

# ERP Software Design for Walmart with Python (Relational Databases & GUI Development)

Design of Walmart Database and GUIs (Store Record Manager, Average Store Size Calculations and Weekly Sales Plots)

**OGHENERUME EFEDUMA**

## Contents

1. Introduction.....	2
1.1 Design Objectives.....	2
1.1.1 Database Design objectives .....	2
1.1.2 Graphical User Interface (GUI) Design objectives .....	2
2. Database Design .....	2
2.1 Database Structure (ERD Diagram) .....	2
2.2 Table Descriptions .....	3
2.3 Database Design Methodology.....	5
2.4 Normalization: .....	5
2.4.1 1 <sup>st</sup> Normal Form .....	5
2.4.2 2 <sup>nd</sup> Normal Form .....	6
2.4.3 3 <sup>rd</sup> Normal form .....	8
2.5 Degrees of Relationships .....	8
2.6 SQL Schema for Creation of Tables in the Database .....	9
3. Data Cleaning .....	11
4. GUI Design .....	13
4.1 Manager and Store Record Management GUI .....	13
4.1.1 Updating Manager Details.....	13
4.2 Average Store Size Calculator and Weekly Sales Plot.....	15
4.2.1 Store Size Calculation:.....	15
4.2.2 Plot Weekly Sales for store and department.....	16
5. Proposed Amazon Web Services (AWS) application for this project.....	18
5.1 Executive Summary .....	18
5.2 Proposed Solution .....	18
5.3 Pricing.....	19
5.4 Benefits.....	19
5.5 Some Technical Issues .....	19
6. References.....	20

## 1. Introduction

This report details the process and results for the design of a hypothetical enterprise software for Walmart consisting of:

- A normalized relational database structured for business needs and developed using sqlite3
- A GUI designed with python Tkinter with three key functionalities:
  - Store & manager records management
  - Average store size calculation by store type
  - Analysis of weekly sales by store and department

A sample dataset with information related to some Walmart stores has been provided and will be cleaned, transformed, and loaded into the database. This data will also be queried from the database using the GUI.

### 1.1 Design Objectives

#### 1.1.1 Database Design objectives

- **Business Requirements:** Database Schema is developed to meet business requirements and functionality of applications
- **Normalization:** The data is organized into separate tables based on their logical relationships, and the tables are normalized to minimize data redundancy and improve data integrity.
  - **Reduced redundancy** - A data field should not appear in multiple tables and schemas in our database and should only exist where needed in business context. If data changes, it should not need to be changed in multiple places to maintain accuracy of all records in our database
  - **Data integrity** - The data in the database follows specific rules and constraints, and that it is free from errors or inconsistencies. Entity integrity maintained by unique identifiers for all records and referential integrity maintained by foreign keys
- **Consistency:** The database enforces constraints and rules to ensure that the data is always accurate and consistent.
- **Data Independence:** The database is designed in such a way that changes to the data structure do not affect the application that uses it.

#### 1.1.2 Graphical User Interface (GUI) Design objectives

- **User Experience:** GUI should be intuitive, well organized, and easy to use by target audience within the enterprise
- **Accessibility:** The database can be easily accessed and queried by users and through the Graphical User Interface and within defined functionality
- **Data Validation:** GUI should be designed for data validation of user inputs e.g. email address inputs must be validated to meet the required format.

## 2. Database Design

### 2.1 Database Structure (ERD Diagram)

The Database will consist of 5 tables derived from the data provided (Kaggle, 2014) and designed to meet the application requirements. See entity relationship diagram below:

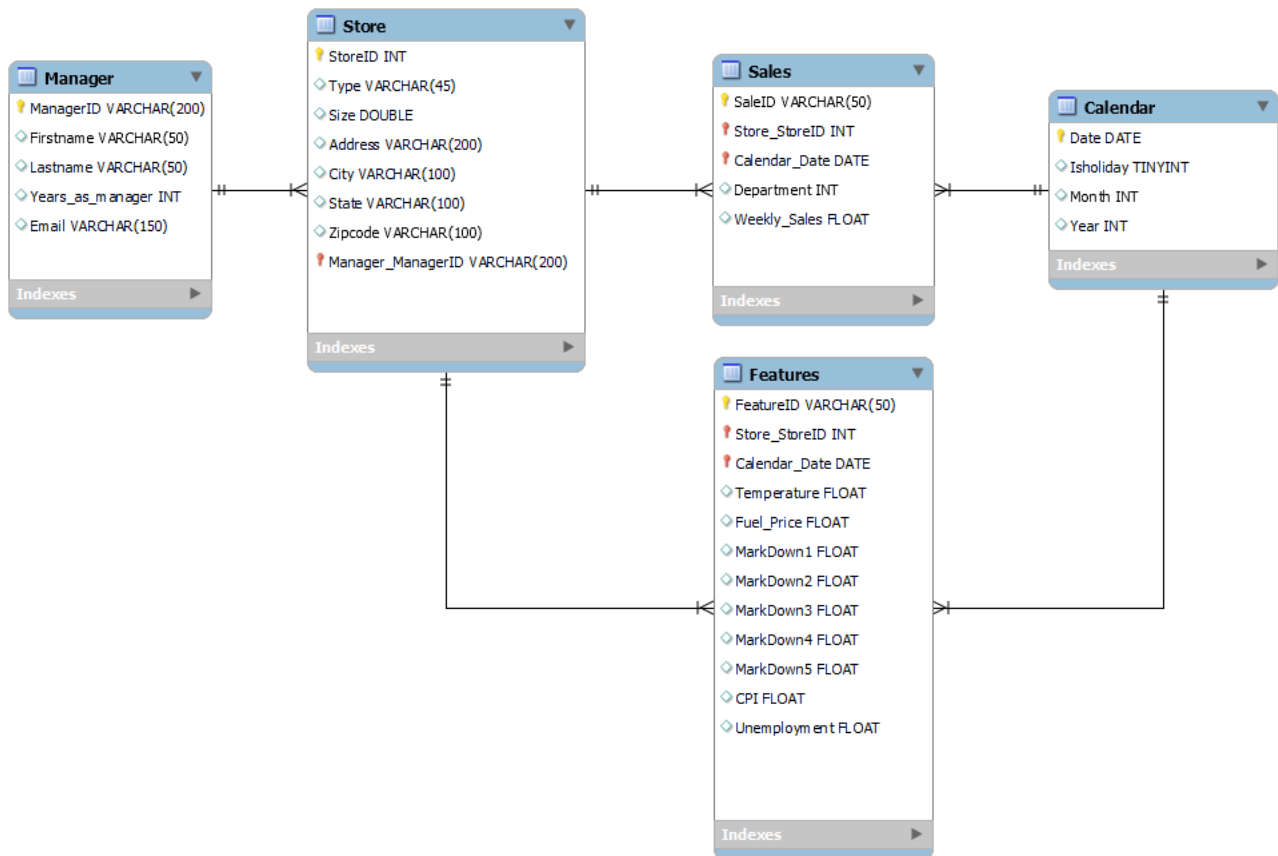


Figure 1: ERD Diagram for Walmart Database designed in MySQL Workbench and using crow foot notation (Oracle, 2023)

## 2.2 Table Descriptions

### 2.2.1 Sales Table

Table 1: Sales Table

Field Name	Data Type	Unit	Constraints	Description
<b>SaleID</b>	Varchar(50)	N/A	Primary Key	Unique identifier for every weekly sales record that identifies the unique combination of the store, department, and date(week) for which the sales were aggregated Example: <b>WS4-3-20100205</b>
<b>StoreID</b>	Integer	N/A	Foreign Key Not Null	Unique identifier for each store in the database, referenced from the Store table
<b>Date</b>	Date	N/A	Foreign Key	Date value representing the week for which the sales were aggregated
<b>Department</b>	Integer	N/A	Not Null	Integers that identify the department for which the sales were aggregated
<b>Weekly_sales</b>	Float	US Dollars (\$)	N/A	Sales aggregated for the week for a department and store in US Dollars

### 2.2.2 Features Table

Table 2: Features Table

Field Name	Data Type	Unit	Constraints	Description
<b>FeatureID</b>	Varchar(50)	N/A	Primary Key	Unique identifier for every feature record that identifies the unique combination of the store and date(week) for which the features were measured and aggregated. Example: <b>F1-20100205</b> .
<b>StoreID</b>	Integer	N/A	Foreign Key Not Null	Unique identifier for each store in the database, referenced from the Store table
<b>Date</b>	Date	N/A	Foreign Key	Date value representing the week for which the aggregated features were recorded.
<b>Temperature</b>	Float	Degrees Fahrenheit	N/A	Average temperature in the region measured for that store and week
<b>Fuel_Price</b>	Float	USD (\$)/gallon	N/A	Fuel price aggregated for the week
<b>Markdown1-5</b>	Float	USD (\$)	N/A	Anonymized data related to promotional markdowns that Walmart is running
<b>CPI</b>	Float	N/A	N/A	CPI recorded during the week
<b>Unemployment</b>	Float	Percent (%)	N/A	Unemployment rate in the US for the week

### 2.2.3 Store Table

Table 3: Store Table

Field Name	Data Type	Unit	Constraints	Description
<b>StoreID</b>	Integer	N/A	Primary Key	Unique identifier for each store in the database
<b>Type</b>	Char(1)	N/A	N/A	Walmart store type denoted by either of three single characters; A, B or C.
<b>Size</b>	Float	Square feet	N/A	Size of Walmart store
<b>Address</b>	Varchar(200)	N/A	N/A	Street address where store is located
<b>City</b>	Varchar (100)	N/A	N/A	City in which store is located
<b>State</b>	Varchar(100)	N/A	N/A	State in which store is located
<b>Zipcode</b>	Varchar(50)	N/A	N/A	Zip code of store location
<b>Manager</b>	Varchar(200)	N/A	Foreign Key	Unique identifier for Walmart store managers

## 2.2.4 Manager Table

Table 4: Manager Table

Field Name	Data Type	Unit	Constraints	Description
<b>ManagerID</b>	Varchar(200)	N/A	Primary Key	Unique identifier for each store manager in Walmart assigned when they become managers and by virtue of this role. Example: <b>robert.alvey-WM22</b>
<b>Firstname</b>	Varchar(50)	N/A	N/A	Manager's first name
<b>Lastname</b>	Varchar(50)	N/A	N/A	Manager's last name
<b>Years_as_manager</b>	Integer	years	Max: 62	Years for which manager has been a manager. This can be a maximum of 62. (Assuming a minimum start age of 18 and max retirement age of 80)
<b>Email</b>	Varchar(200)	N/A	N/A	Email address of manager

## 2.2.5 Calendar Table

Table 5: Calendar Table

Field Name	Data Type	Unit	Constraints	Description
<b>Date</b>	Date	N/A	Primary Key	Unique Date values representing the weeks in the year
<b>Isholiday</b>	Boolean(Tinyint)	N/A	N/A	<b>TRUE (1)</b> if the week is a special holiday week and <b>FALSE(0)</b> if not
<b>Month</b>	Integer	N/A	N/A	The month of the date
<b>Year</b>	Integer	N/A	N/A	The year of the date

## 2.3 Database Design Methodology

The key considerations for the database design were

- **Normalization:** Application of Normal Forms up to the third normal form (3NF)
- **Degrees of relationship:** Maintaining one to many relationships

## 2.4 Normalization:

### 2.4.1 1<sup>st</sup> Normal Form

**Rule/Check:** Each column in our database contains just one value (atomic values)

- Manager name split into firstname and lastname (McNeille, 2022) (Wikipedia Foundation, Inc., 2023) (Microsoft, 2022).

Manager	FirstName	LastName
Robert Alvey	Robert	Alvey
Jerry Martinez	Jerry	Martinez
Susanna Kellner	Susanna	Kellner

Figure 2: Manager name split into firstname and lastname

Address	Street Address	City	State	Zipcode
4971 Janet Court;Livermore;CA;94550	4971 Janet Court	Livermore	CA	94550
4439 Gale Street;Livermore;CA;94550	4439 Gale Street	Livermore	CA	94550
856 Milton Street;Oakland;CA;94607	856 Milton Street	Oakland	CA	94607

Figure 3: Store address broken down into components

**Rule/Check:** No repeating groups within any table in the provided dataset (ref)

- No repeating groups identified within the data provided

**Rule/Check:** Separate table for each set of related data and each table must represent a single subject (McNeille, 2022) (Wikimedia Foundation, Inc., 2023) (Microsoft, 2022).

- Store, Sales and Features tables created from data

Table 6: Separate tables created for entities and related data

Entity/Table	Related Fields
<b>Store</b>	Store, Type, Size, Street Address, City, State, Zipcode, Manager, Years as manager, email
<b>Sales</b>	Store, Department, Date, Weekly sales, Isholiday
<b>Features</b>	Store, Date, Temperature, Fuel Price, Markdown(1-5), CPI, Unemployment, Isholiday

- The **Department** is also a separate entity identified but with no related data or descriptive information to merit the creation of another table. It also appears only in the sales table and is not referenced elsewhere. The field is thus maintained as an attribute in the sales records in this scenario.

**Rule/Check:** Identify each set of related data with a primary key (McNeille, 2022) (Wikimedia Foundation, Inc., 2023) (Microsoft, 2022).

- **Composite key [store + email]** identified for the **Store table** – All other fields in each record in the store table can be identified by a part of the composite key. The email is chosen as it is the only unique identifier for the store manager.
- **Composite key [store + department + date]** identified for the **Sales table**: All other fields in each record in the Sales table can be identified by a part of the composite key.
- **Composite key [store + date]** identified for **Features table** : All other fields in each record in the Sales table can be identified by a part of the composite key.

#### 2.4.2 2<sup>nd</sup> Normal Form

**Rule/Check:** Can we figure out “any of the values in a row” from just part of the composite/compound key?

- In the store table, we can figure out the manager’s **firstname, lastname, and years as manager** once we identify a manager based on his email which is only a part of the composite key. Hence, we create a separate table for the manager using the email as a foreign key in the store table.

Table 7: Separate table created for Manager

Entity/Table	Fields
Store	Store, Type, Size, Street Address, City, State, Zipcode, <b>email(Manager)</b>
<b>Manager</b>	<b>Manager, Years as manager, email</b>

- In the sales table with the composite key of store + department + date, all attributes in this table depend on all parts of the composite key except **Isholiday** which depends only on the **date**. To conform to 2NF and remove duplicities, every non-key attribute must depend on the whole primary key. A separate table (Calendar) for Date values and it’s dependent attribute, **‘Isholiday’** is created while the date is referenced as a foreign key in the sales table.
- A similar issue exists with the features table with the composite key of store + date, where **‘Isholiday’** can be determined only from the date column of the key. Hence the same

justification applies for the creation of the Calendar table with the date referenced as a foreign key in the features table.

(McNeille, 2022) (Wikimedia Foundation, Inc., 2023) (Microsoft, 2022)

Table 8: Calendar table created for date values

Entity/Table	Fields
Sales	Store, Department, <b>Date</b> , Weekly sales
Features	Store, <b>Date</b> , Temperature, Fuel Price, Markdown(1-5), CPI, Unemployment,
<b>Calendar</b>	<b>Date, Isholiday</b>

**Rule/Check:** According to (Microsoft, 2022), to fulfil the 2NF, we need to create a separate table for sets of values that apply to multiple records.

- The date and it's associated attribute 'Isholiday,' appears in the sales and features tables. This further justifies making a separate table for Date values and relating it to the features and sales table with Date as a foreign key.
- The store table appears in features and sales tables and is related to these tables with a foreign key.

**Rule/Check:** If a table has a single column primary key, it automatically satisfies 2NF, but if a table has a multi-column or composite key then it may not satisfy 2NF (Microsoft, 2022).

- Store table records will be identified with **StoreID** integer primary key
- Calendar table records will be identified with **date** as the primary key
- In determining a potential primary key for records in our Manager table, the most unique information we have for the manager is their email which is unique for every employee. The context of this key is enhanced by making a unique identifier for each store manager in Walmart assigned to them when they become managers. Example: **robert.alvey-WM12**. The first element "**robert.alvey**" is from the prefix of the manager's email (a unique attribute for all employees) and "**WM12**" represents the added context of Walmart Manager, promoted or hired into managerial position in 2012 (this points to years served as a manager).

Table 9: ManagerID created for Manager table

ManagerID	firstname	lastname	years_as_manager	email
<b>robert.alvey-WM22</b>	Robert	Alvey	1	robert.alvey@walmart.org
<b>jerry.martinez-WM12</b>	Jerry	Martinez	11	jerry.martinez@walmart.org
<b>susanna.kellner-WM16</b>	Susanna	Kellner	7	susanna.kellner@walmart.org
<b>marco.spivey-WM22</b>	Marco	Spivey	1	marco.spivey@walmart.org
<b>timothy.narvaez-WM10</b>	Timothy	Narvaez	13	timothy.narvaez@walmart.org

- A distinct and unique Primary key identifier (**SaleID**) is developed based on the composite key. Example: **WS4-3-20100205**. The first element '**WS4**' represents weekly sales for **storeID 4**. The second element '**3**' points to the **department** and the last element is the date which in this example is **5<sup>th</sup> February 2010**. This will also enable the records to be referenced as a foreign key other tables which can't be done with a composite key.



Table 10: SaleID created for Sales Table

saleID	storeID	department	date	weekly_sales
<b>WS3-1-20100205</b>	3	1	05/02/2010	24924.5
<b>WS5-1-20100212</b>	5	1	12/02/2010	46039.49
<b>WS7-2-20100219</b>	7	2	19/02/2010	41595.55

- A distinct and unique primary key identifier (**FeatureID**) is developed based on the composite key. Example: **F1-20100205**. The first element '**F1**' represents features measured for **storeID 1**. The second element '**20100205**' points to the date(week) of measurement which in this example is the **5<sup>th</sup> February 2010**. This will enable the records to be referenced as a foreign key in other tables.

Table 11: Feature ID created for feature table

featureID	storeID	date	temperature	fuel_price	...
<b>F1-20100205</b>	1	40214	42.31	2.572	...
<b>F1-20100212</b>	1	40221	38.51	2.548	...
<b>F1-20100219</b>	1	40228	39.93	2.514	...

### 2.4.3 3<sup>rd</sup> Normal form

**Rule/Check:** Can we figure out "any of the values in a row" from any of the other values in the same row? (McNeille, 2022) (Wikimedia Foundation, Inc., 2023) (Microsoft, 2022)

- In the store table, a chain of transitive dependencies exists where the zipcode can be determined from the street address, the city can be determined from the zipcode and the state can be determined by the city. This points to a separate table being created for the address records.
- Also, in a broader scenario, these store addresses could be referenced in multiple tables such as tables with sales records, procurement records and supply chain records. Nonetheless, the database design requirements in this scenario does not require this. Hence, the street address (and its related data) is maintained as an attribute of the store entity.
- Besides the instance mentioned above the database up to this point adheres to the checks of the 3NF

## 2.5 Degrees of Relationships

In addition to normalization we considered the following assumptions and relationships

### **Manager - Store (One-Many)**

- We assume A manager could be assigned as a referent for more than one store (maybe 2 small-sized Walmart stores). Hence, a one to many relationship between manager and store is considered with the managerID referenced as a foreign key in the store table

### **Store - Features, Store - Sales (One-to-many)**

- The stores and features tables contain records for multiple stores and for multiple dates which implies multiple occurrences of a specific store in these tables. The store is thus referenced in the sales table and in the features table as a foreign key. Hence a one-to-many relationship was considered in both instances.

**Date - Features, Date - Sales (One-to-many)**

- The sales and features tables have records for specific dates and for multiple stores which implies multiple occurrences of a specific date in these tables. Hence a one to many relationship was considered in both instances (1:N)

**Other considerations**

Each store has only 1 address. Hence creating an address table will result in a One-to-One relationship between store and address tables. Considering no other tables reference store addresses in this case, it will be maintained as an attribute of the store in this scenario.

## 2.6 SQL Schema for Creation of Tables in the Database

### 2.6.1 Sales Table

```
create_sales = """
CREATE TABLE IF NOT EXISTS Sales(
saleID VARCHAR(50),
storeID INTEGER NOT NULL,
department INTEGER NOT NULL,
date DATE,
weekly_sales FLOAT,
PRIMARY KEY (saleID),
FOREIGN KEY (storeID) REFERENCES Store(storeID),
FOREIGN KEY (date) REFERENCES Calendar(date));
"""
```

### 2.6.2 Features Table

```
create_features = """
CREATE TABLE IF NOT EXISTS Features(
featureID VARCHAR(50),
storeID INTEGER NOT NULL,
date DATE,
temperature FLOAT,
fuel_price FLOAT,
markdown1 FLOAT,
markdown2 FLOAT,
markdown3 FLOAT,
markdown4 FLOAT,
markdown5 FLOAT,
CPI FLOAT,
unemployment FLOAT,
PRIMARY KEY (featureID),
FOREIGN KEY (storeID) REFERENCES Store(storeID),
FOREIGN KEY (date) REFERENCES Calendar(date));
"""
```

### 2.6.3 Store Table

```
create_store = """"  
  
CREATE TABLE IF NOT EXISTS Store(  
storeID INTEGER,  
type VARCHAR(45),  
size DOUBLE,  
address VARCHAR(200),  
city VARCHAR(100),  
state VARCHAR(100),  
zipcode VARCHAR(50),  
managerID VARCHAR(200),  
PRIMARY KEY (storeID),  
FOREIGN KEY (managerID) REFERENCES Manager(managerID));  
  
"""
```

### 2.6.4 Manager Table

```
create_manager = """"  
  
CREATE TABLE IF NOT EXISTS Manager(  
managerID VARCHAR(200),  
firstname VARCHAR(50),  
lastname VARCHAR(50),  
years_as_manager INTEGER,  
email VARCHAR(200),  
PRIMARY KEY (managerID));  
  
"""
```

### 2.6.5 Calendar Table

```
create_calendar = """"  
  
CREATE TABLE IF NOT EXISTS Calendar(  
date DATE,  
isholiday BOOLEAN,  
month INTEGER,  
year INTEGER,  
PRIMARY KEY (date))  
  
"""
```

### 3. Data Cleaning

#### 3.1 Data Cleaning Steps

##### 3.1.1 Store Table

Total records: 45

- Store table created in line with Database schema
- Replaced missing store size values with the average size of the corresponding store type


Type	Average Size		Store	Type	Size
A	178425.571		15	B	<b>99781.5625</b>
B	<b>99781.5625</b>		26	A	<b>40541.66667</b>
C	<b>40541.66667</b>				

Figure 4: Replaced missing 'size' values with average size based on store type, where applicable


- Replaced missing managerIDs in store table with 'unassigned'

##### 3.1.2 Manager Table

Manager table created in line with Database design with 'ManagerID' as foreign key Store table

- Replaced unrealistic 'years\_as\_manager' value with mean value of 8 years after outlier removed. 1 record observed with 100 years.

ManagerID	First_name	Last_name	Years_as_manager	Email
William.Rodriguez-WM15	William	Rodriguez	<b>100</b>	william.rodriguez@walmart.org




ManagerID	First_name	Last_name	Years_as_manager	Email
William.Rodriguez-WM15	William	Rodriguez	<b>8</b>	william.rodriguez@walmart.org

Figure 5: Replace unrealistic 'years\_as\_manager' value with mean value

- Removed null values from managerID and drop duplicates
- Filled in missing first and last name values using email prefix where email is available [George Parker]

Store	First_name	Last_name	...	Email
4	Marco	Spivey	...	Marco.Spivey@Walmart.org
5	Timothy	Narvaez	...	Timothy.Narvaez@Walmart.org
6			...	George.Parker@Walmart.org



Store	First_name	Last_name	...	Email
4	Marco	Spivey	...	Marco.Spivey@Walmart.org
5	Timothy	Narvaez	...	Timothy.Narvaez@Walmart.org
6	<b>George</b>	<b>Parker</b>	...	<a href="mailto:George.Parker@Walmart.org">George.Parker@Walmart.org</a>

Figure 6: Filled in missing first and last name values using email prefix

### 3.1.3 Features Table

- Features table created in line with Database design with unique identifier ('FeatureID') as Primary key
- Re-organized columns in features table to fit database schema
- Filled in the missing CPI values with the average CPI value for the corresponding Store
- Filled in the missing Unemployment values with the average unemployment value for the corresponding year

### 3.1.4 Date Table

- Date table created in line with database design with Date as foreign key in Sales and Features tables. 'Isholiday' dropped from Sales and Features tables.
- Merge date values from features and sales tables and remove duplicates to prepare data for loading into the Calendar table of the database with date as primary key

## 4. GUI Design

### 4.1 Manager and Store Record Management GUI

The GUI is designed to enable the update of a manager's records in the database

#### 4.1.1 Updating Manager Details

1. Input Store ID, click "Show store Data"
2. Store and manager details for specific store retrieved from database and displayed in GUI
3. Click "Update Manager Data" to open "Updating Manager Details" window. Here manager information can be modified and saved to the database
4. Attempt to input email in wrong format: [robert.orwig@wal.org](mailto:robert.orwig@wal.org)
5. Error prompt comes up when user attempts to save manager information with wrong email format

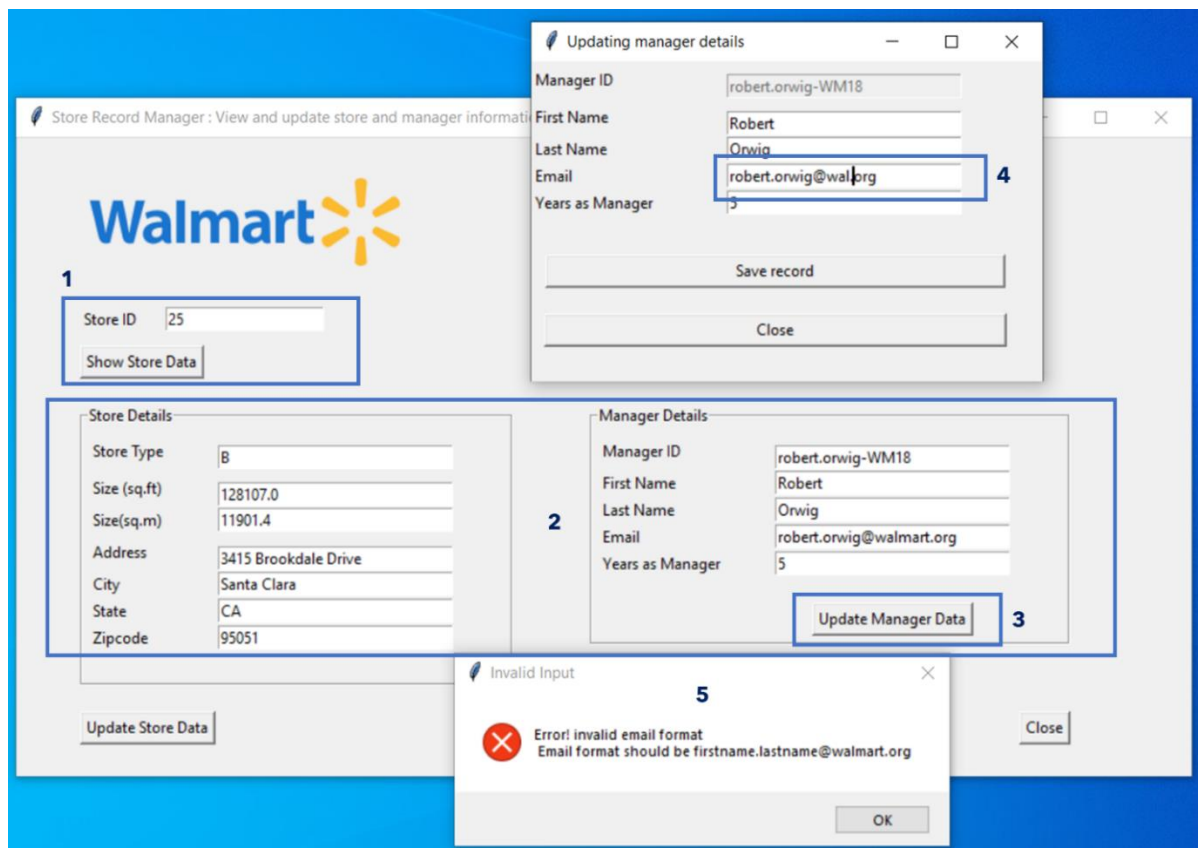


Figure 7: Update manager email. Data validation to ensure correct email format

6. Input valid Manager details
7. Click "Save record" to save changes to database
8. Prompt appears confirming Manager details updated

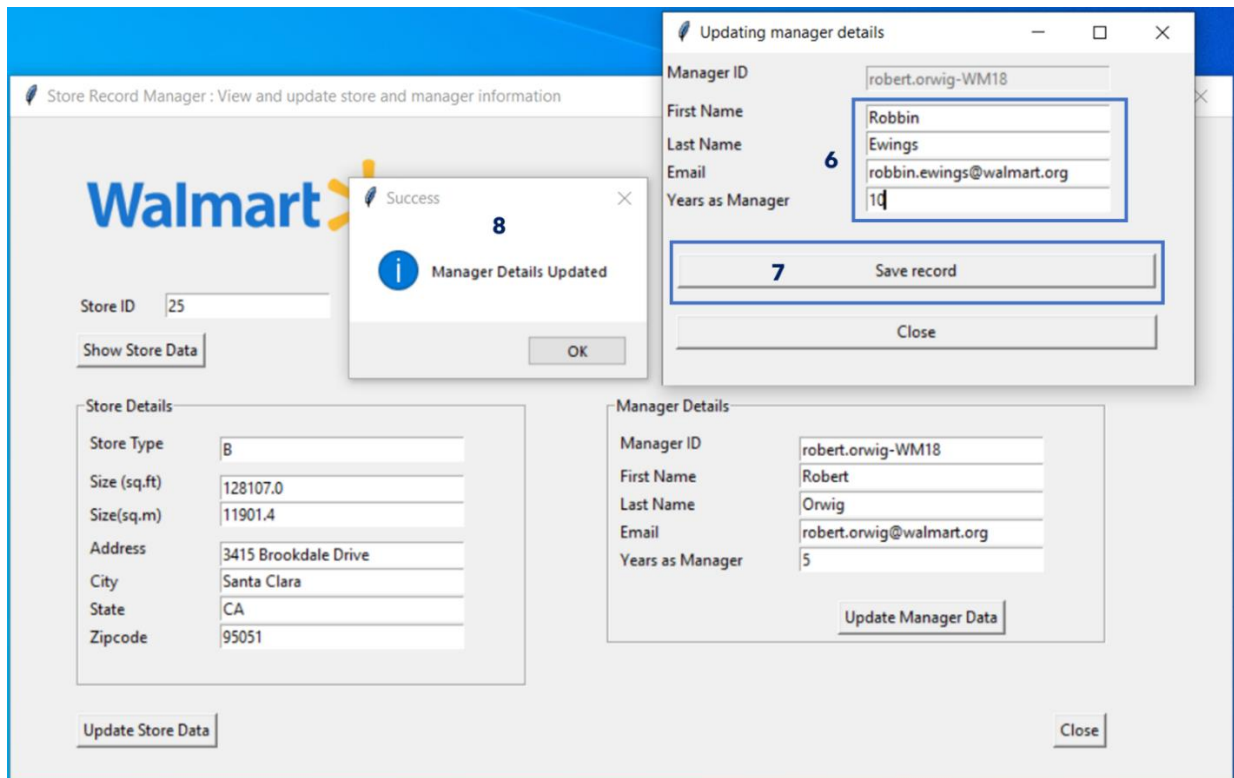


Figure 8 Updating manager details. Successful update

9. Input Store ID, click "Show store Data" to reveal updated details for Store 25
10. Updated manager details for store 25 retrieved from database and displayed in GUI
  - First name changed from '**Robert**' to '**Robbin**'
  - Last name changed from '**Orwig**' to '**Ewings**'
  - Years a manager changed from **5** to **10**

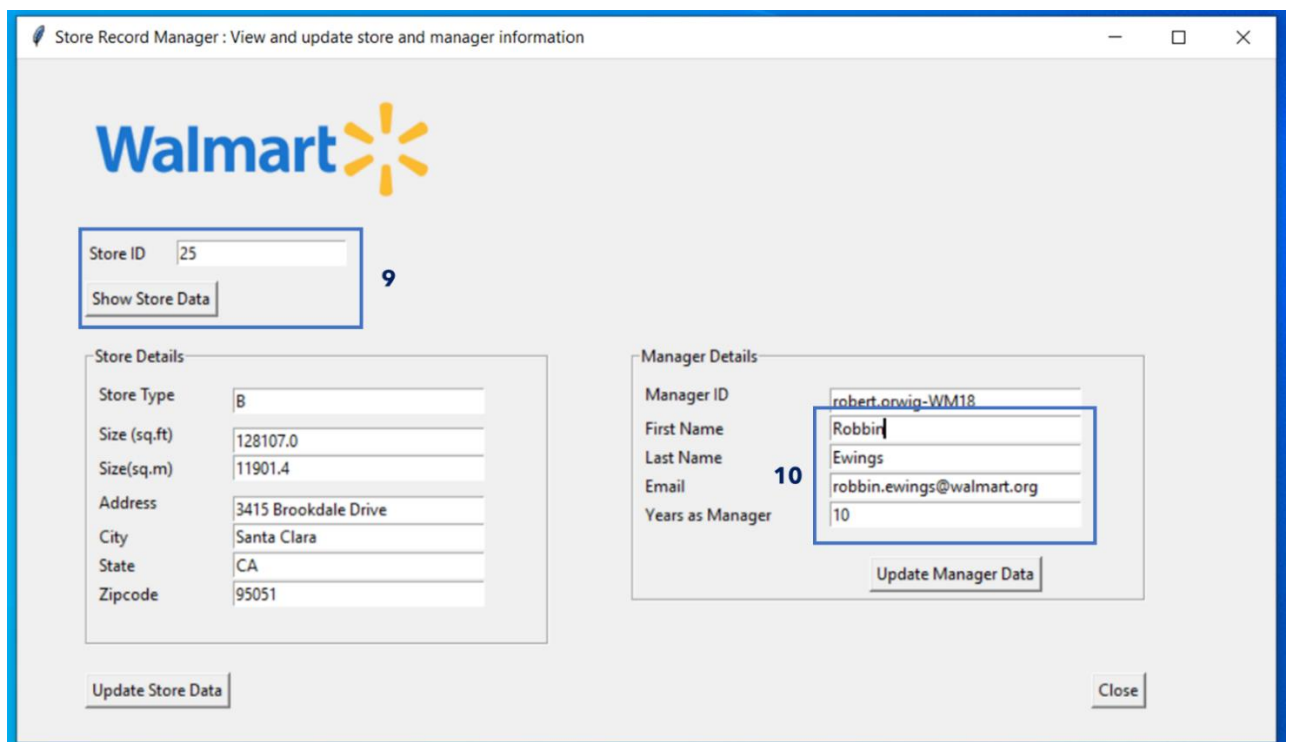


Figure 9 Check for updated manager details

11. Click "Update store Data" to open window for store data updates
12. Updated the database with Type, Size, Address, City, State, Zipcode for the specific store
13. Assign a different manager to the store by inputting a new Manager ID in store details. Only managers in database can be assigned to a store.

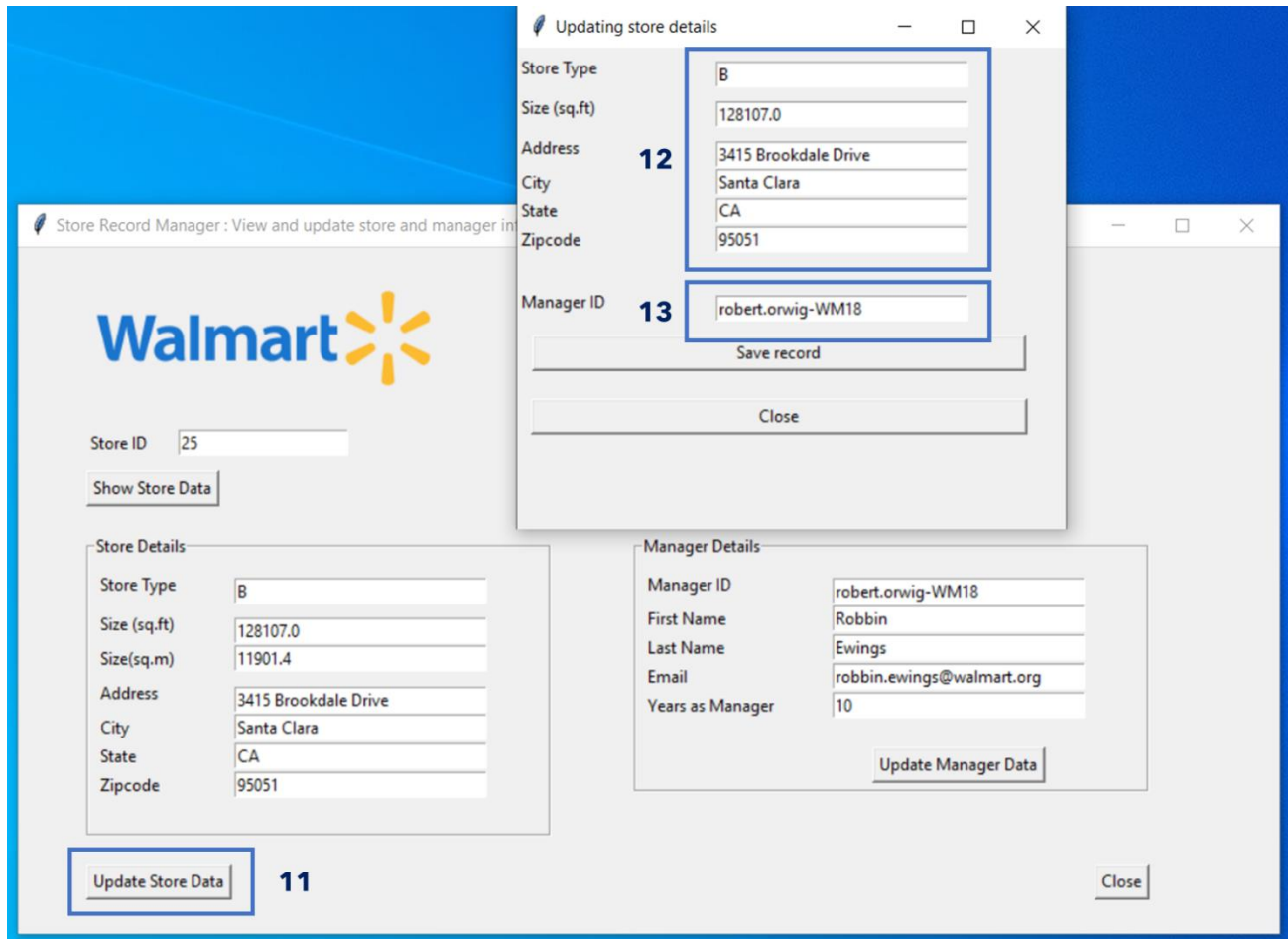


Figure 10 Update store details. Assign another manager to store

## 4.2 Average Store Size Calculator and Weekly Sales Plot

### 4.2.1 Store Size Calculation:

1. Click "Show sizes button"
2. Average store size by store type in square feet retrieved from database and displayed in GUI
3. Message prompt confirming values have been recalculated from database



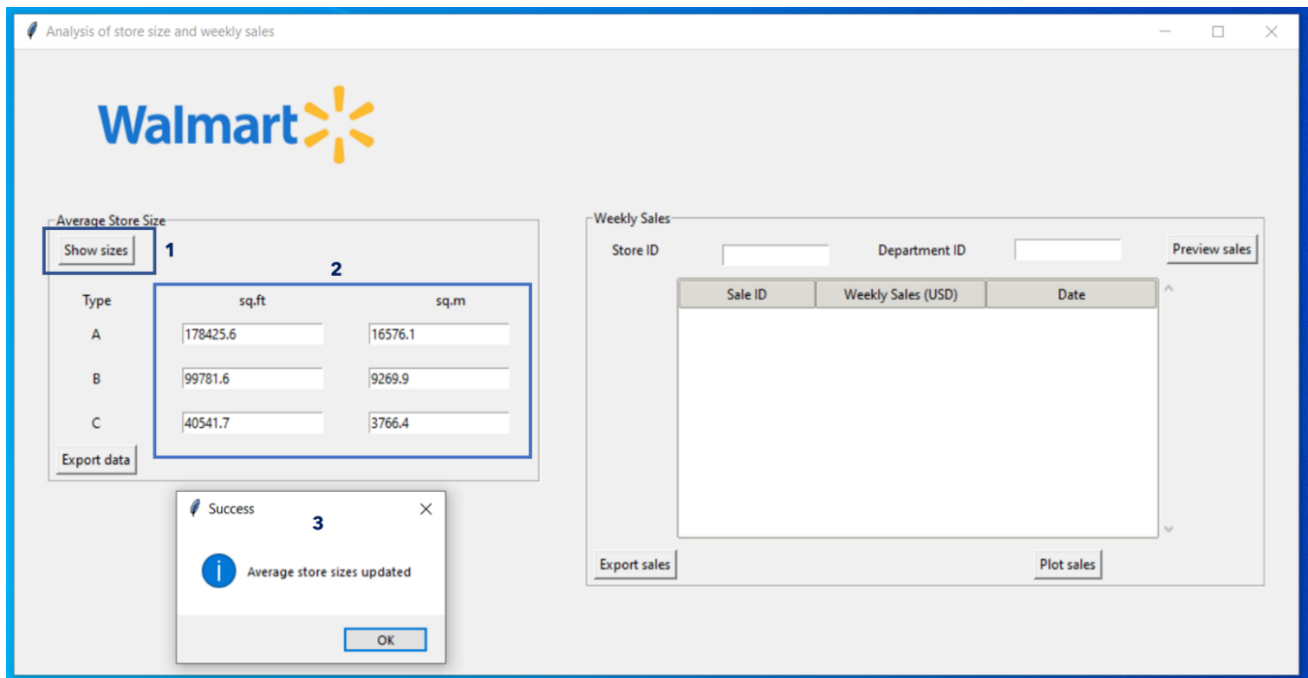


Figure 11: Store Size Calculation

#### 4.2.2 Plot Weekly Sales for store and department

##### Steps:

1. Input store and department (Input validation ensures values are integers and exist in the database e.g. Inputting 96 in the Store ID entry box will throw an error because no store with that store ID exists in the database)
2. Click "Preview sales"
3. Preview of historical weekly sales for selected store and department generated (1<sup>st</sup> 100 records only)
4. Click "Plot sales button" to plot weekly sales over time
5. Plot of weekly sales by date for store and department (N.B: Sales plot can be generated even without preview)

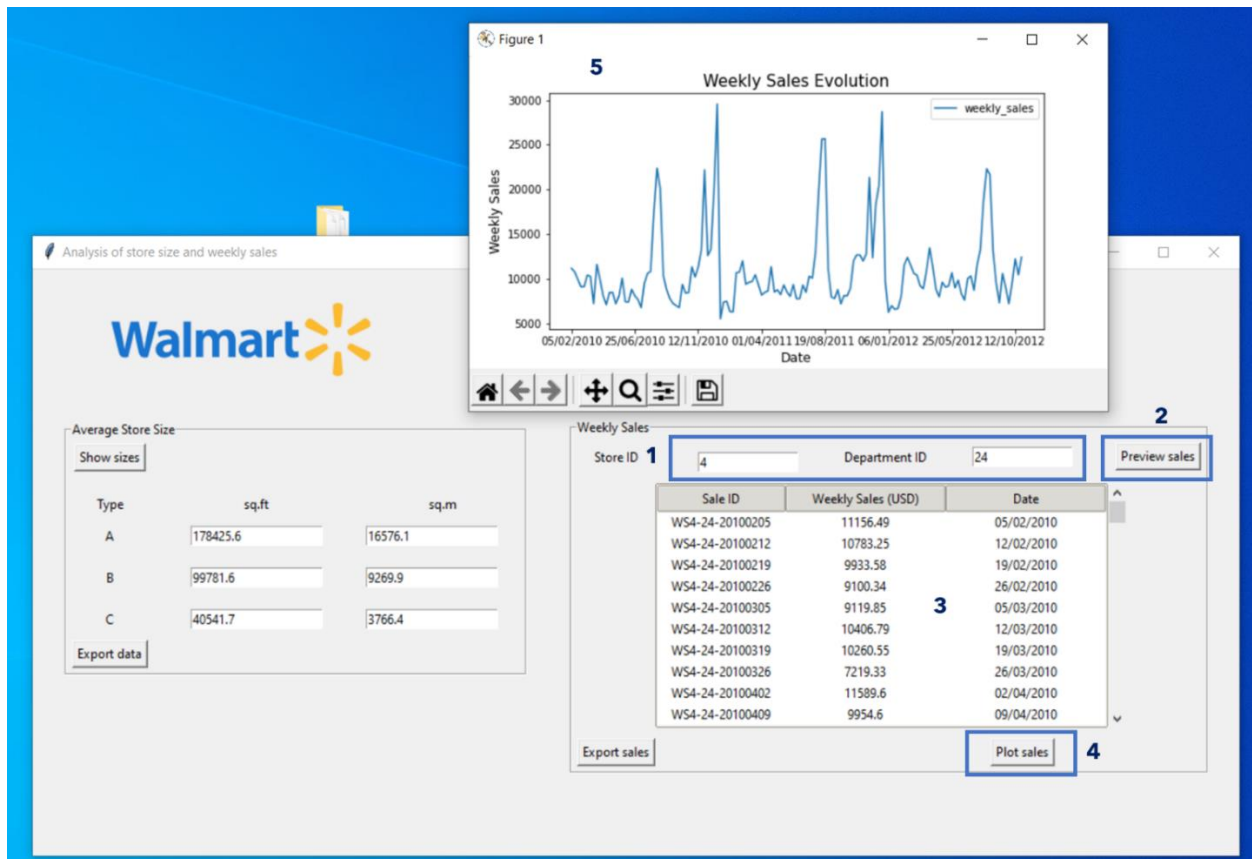


Figure 12: Weekly Sales plot for specific store and department

## 5. Proposed Amazon Web Services (AWS) application for this project

### 5.1 Executive Summary

Walmart Inc., the world's largest retailer according to Statista (Ozbun. 2022), understand the digital transformation initiative that involves data platforms and desire to predict sales in different departments from various factors: such as the temperature, whether there is a holiday in specific week, the unemployment rate and markdown on the prices respectively.

With on premise environments having its drawbacks, the cloud eliminates the trouble of maintaining and updating systems due to its flexibility, dependability, and security, allowing you to devote your time, money, and resources to realizing key business strategy (Xperience, 2017). In the next five years, the vast majority of firms (93%) will either fully migrate to the cloud or use a combination of cloud and on-premise solutions (Macrae, 2022).

Amazon Web services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally (AWS, 2023). With no long-term contracts or up-front commitments, AWS is an easy- to-use, flexible, scalable, cost effective and secure PaaS (Platform-as-a-service. This makes it a solid choice to host the MySQL database for Walmart. (AWS, 2023)

### 5.2 Proposed Solution

A group of managed services known as **Amazon Relational Database Service** (Amazon RDS) makes it simple to set up, run, and scale databases (MySQL in this case) in the cloud (AWS, 2023)

While automating time-consuming administrative activities like hardware provisioning, database setup, patching, and backups, it offers affordable and expandable capacity. Giving applications the quick performance, high availability, security, and compatibility, they require, and this frees you to concentrate on the platform you want to build (AWS, 2022)

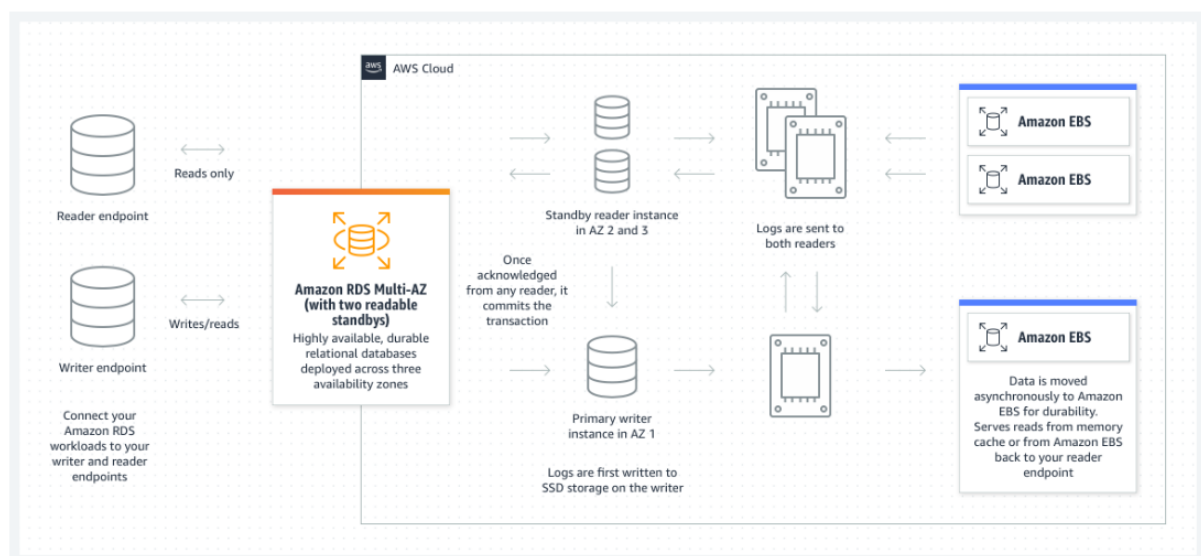


Figure 13: Amazon RDS Multi-AZ with two readable standbys logical architecture (AWS, 2023)

Utilizing Amazon RDS Multi-AZ with two readable standbys, deploy robust, highly available MySQL or PostgreSQL databases in three AZs. Gain automatic failovers that take place in less than 35 seconds on average, up to 2x quicker transaction commit latency than Amazon RDS Multi-AZ with one standby, more read capacity, and the option to run computation on either AWS Graviton2 or Intel-based instances (AWS, 2023).

### 5.3 Pricing

Using AWS Calculator, below is the Monthly cost for Amazon RDS.

Service Name	Upfront Cost	Monthly cost (USD)	Total 12 Months Cost (USD)
Amazon RDS for MySQL	0.00	389.34	4,672.08

**Please note: This pricing is exclusive of VAT.**

### 5.4 Benefits

- With Amazon RDS, you may access the features of databases like MariaDB, MySQL, SQL Server, Oracle, or PostgreSQL; thus, all of the programmes, tools, and code you now use with your current database should function properly.
- This maintains database software up to date by automatically handling updates and backups through Amazon RDS. The ability to scale the amount of storage or computing power attached to your relational database instance is advantageous to developers.
- Replication is now easier to utilise thanks to Amazon RDS. Replication can be used to grow beyond the capabilities of a single database instance, which is required for read-heavy database workloads, improve database availability, and enhance data durability.
- No up-front costs and you only pay what you utilize.
- Advanced data protection features offered by Amazon RDS include multi-factor authentication, automated data encryption at rest, SSL/TLS for data encryption in transit, and activity monitoring using AWS CloudTrail.

(Maayan, 2022)

### 5.5 Some Technical Issues

1. Quotas in RDS: Each AWS account has quotas, for each AWS Region, on the number of Amazon RDS resources that can be created. After a quota for a resource has been reached, additional calls to create that resource fail with an exception
2. There are naming constraints in RDS
3. There is a maximum limit for database connections: For MySQL – “1-100000”
4. There are file size limits: the maximum provisioned storage limit constrains the size of a table to a maximum size of 16 TB. Lower versions of MySQL have different conditions that apply to them as far as this constraint is concerned.

(AWS, 2023)

## 6. References

- Wikimedia Foundation, Inc., 2023. *Database normalization*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Database\\_normalization](https://en.wikipedia.org/wiki/Database_normalization)  
[Accessed 16 January 2023].
- Amazon Web Services, Inc. , 2023. *What is AWS?*. [Online]  
Available at: [https://aws.amazon.com/what-is-aws/?nc1=f\\_cc](https://aws.amazon.com/what-is-aws/?nc1=f_cc)  
[Accessed 15 January 2023].
- AWS, 2022. *Overview of Amazon Web Services - AWS Whitepaper*. [Online]  
Available at: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>  
[Accessed 15 January 2023].
- AWS, 2023. *Amazon RDS*. [Online]  
Available at: <https://aws.amazon.com/rds/>  
[Accessed 15 January 2023].
- AWS, 2023. *Amazon Relational Database Service User Guide*. [Online]  
Available at:  
[https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP\\_Limits.html#RDS\\_Limits\\_Constraints](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Limits.html#RDS_Limits_Constraints)  
[Accessed 18 January 2023].
- Consortium, S., 2022. *Datatypes In SQLite*. [Online]  
Available at: <https://www.sqlite.org/datatype3.html>  
[Accessed 4 January 2023].
- Kaggle, 2014. *Walmart Recruiting - Store Sales Forecasting: Dataset Description*. [Online]  
Available at: <https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/data>  
[Accessed 20 December 2022].
- Maayan, G., 2022. *Getting started with MySQL on AWS: Costs, Setup, and Deployment options*. [Online]  
Available at: <https://www.sqlshack.com/getting-started-with-mysql-on-aws-costs-setup-and-deployment-options/>  
[Accessed 18 January 2023].
- MacRae, D., 2022. *93% of IT industry to adopt cloud tech within five years*. [Online]  
Available at: <https://www.cloudcomputing-news.net/news/2022/mar/15/93-of-it-industry-to-adopt-cloud-tech-within-five-years/>  
[Accessed 15 January 2023].
- McNeile, C., 2022. *COMP5000: Software Development and Databases (Lecture Note 3, 5, 7, 8 & 9)*. Plymouth: University of Plymouth.
- Microsoft, 2022. *Description of the database normalization basics*. [Online]  
Available at: <https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description>  
[Accessed 20 December 2023].
- Oracle, 2023. *MySQL Enterprise Edition*. [Online]  
Available at: <https://www.mysql.com/products/workbench/>  
[Accessed 16 January 2023].

ProjectPro, 2023. *How Big Data Analysis helped increase Walmarts Sales turnover?*. [Online]  
Available at: <https://www.projectpro.io/article/how-big-data-analysis-helped-increase-walmarts-sales-turnover/109>  
[Accessed 16 January 2023].

Statista: Ozbun, T., 2022. *Walmart - Statistics & Facts*. [Online]  
Available at: [https://www.statista.com/topics/1451/walmart/#topicHeader\\_\\_wrapper](https://www.statista.com/topics/1451/walmart/#topicHeader__wrapper)  
[Accessed 15 January 2023].

Xperience, 2017. *Cloud Vs On Premise Software: Which Is Best For Your Business?*. [Online]  
Available at: <https://www.xperience-group.com/news-item/cloud-vs-on-premise-software/>  
[Accessed 14 January 2023].