DATABASE MANAGEMENT AND DATABASE DESIGN

**DATABASE DESIGN DOCUMENT**

PROJECT MODULE – 4

BIKE SHARING MANAGEMENT SYSTEM

TEAM MEMBERS

|  |  |
| --- | --- |
| 1 | Ashish Mhatre |
| 2 | Ashwin Kadam |
| 3 | Harshad Jadhav |
| 4 | Dalsi Patel |
| 5 | Samruddhi Narawade |

|  |  |  |
| --- | --- | --- |
| **INDEX** | | |
| **Serial no.** | **Content** | **Page No.** |
| 1. | Business Problem | 2 - 3 |
| 2. | Role Description | 4 |
| 3. | Database Software Utilities | 5 |
| 4. | Business Rules | 6 - 8 |
| 5. | Business Views | 9 |
| 6. | Data Flow Diagram | 10 - 13 |
| 7. | Entity Relationship Diagram | 14 |
| 8. | Table Entity and Attributes | 15 - 25 |
| 9. | Instructions to run the scripts | 26 |
| 10. | Scripts | 27 |

Database Design

**BUSINESS PROBLEM**: - The purpose of the database is to facilitate customer onboarding by creating a rider account, maintain ride details and transactional payment details. Also identify frequent user customers through ride details and offer additional discount offers to increase customer retention. Use ride details to track the health of the bikes and schedule maintenance by notifying maintenance job to maintenance team. Additionally, implement data security for customer and payment details along with generating analytical reports for business development team for management and expansion of business by identifying treads and patterns.

**OBJECTIVES:** -

1. Handle Customer onboarding and offboarding

***Solution:***Maintain Customer and Customer\_Address table to store customer information and use this information to deliver personalized offers as per ride records, Business development team can send out welcome emails by querying this information to direct customer back into service.

1. Track Health of bikes and schedule/Notify maintenance team.

***Solution:*** Maintain Ride\_Transaction table and Bike table to store details on each ride. Utilize this information to calculate total distance travelled by a particular bike and schedule maintenance when a decided threshold distance is reached. Notify maintenance team for maintenance

1. Improve services based on customer’s feedback report

***Solution:*** Maintain Customer\_Feedback table to store customer feedback data and based on this data we will generate the customer ride experience report for the business team.

1. Offer additional discounts and promo offers to daily active users.

***Solution:*** Maintain Ride\_Transaction, Payment and Customer table to store information about customer usage and payment details. Generate detail reports by using riding pattern and payment details to identify frequent service user. Provide a view to business or analytical team for further in-depth analysis.

1. Deliver business insights to build stations at high demand zip codes

***Solution:***  From the Ride\_Transaction table and Station table, generate a view to show the number of bookings with respect to zip codes. This will help the business team to identify the busiest station among the riders.

1. Revenue generated per quarter/month based on zip code/location

***Solution:***  From the Payment, Ride\_transaction and Station table generate a report to sum up the revenue based on the zip code or location based on which we will get information about the performing and non-performing regions.

Role Description:

|  |  |  |
| --- | --- | --- |
| Role | Name | Responsibility |
| DB\_OWNER | MHATREA | DB\_OWNER has permissions to CONNECT, CREATE SESSION and CREATE TABLES. He also has all DDL and DML rights |
| DEVELOPER | KADAMA | DEVELOPER has permissions to CONNECT and CREATE SESSIONS. He has rights to perform operations like INSERT, UPDATE, DELETE and SELECT on all ENTITIES |
| USER\_TESTER | JADHAVH | USER\_TESTER has permissions to CONNECT and CREATE SESSION. He also has permission to SELECT, INSERT and UPDATE on ENTITES – CUSTOMER, CUSTOMER\_ADDRESS, CUSTOMER FEEDBACK. He can only perform SELECT on DISCOUNT ENTITY and can perform SELECT and INSERT on PAYMENT ENTITY |
| QA | JOHNJ | QA has rights to CONNECT and CREATE SESSION. Another permissions TBD. |

Database Software Utilities:

|  |  |  |
| --- | --- | --- |
| Vendor | Product | Function |
| ORACLE CORPORTATION | SQL DEVELOPER | Browse, create, edit, and delete (drop) database objects; run SQL statements and scripts; edit and debug PL/SQL code; manipulate and export data; and view and create reports |

Business Rules:

**Customer Entity**

* Every customer should have a customer id
* Every customer should have a first name
* Every customer should enter appropriate gender
* Every customer should have a unique email id
* Every customer should have a unique contact number
* Every customer should have SSN number
* A customer can be ‘Registered’ or ‘Casual’ customer

**Customer Address Entity**

* Every address should have an address id
* Every customer should have an address and that address could be a ‘Home address’ or ‘Work address’
* Every customer should enter a city
* Every customer should enter a state
* Every customer should enter a zip code
* Each address should have a customer id associated with it

**Bike Entity**

* Every Bike should have a unique Bike ID associated with it.
* Every Bike Status should indicate whether it is Active or Inactive.
* Serviced bikes should have a last service date updated.
* Every Bike should have a commissioned date
* Service bike should mention the employee id of employee who serviced the bike
* Every bike should have a unique body identification number
* Every bike’s distance must be captured and updated against it.
* After completion of 50km for a bike the status should be changed to inactive and bike should be serviced, after service completion bike distance should be reset to 0km and status should be reset to active and service records shall be updated accordingly.

**Station Entity**

* Every station should have a unique Station ID.
* Each Station should have details such as City, Zip code and State.
* Every Station should have a unique Station Name.
* Each Station have a limited specific bike parking capacity.

**Bike at Station Entity**

* Each bike parked at station should have a respective bike id and Station id associated with the entry.

**Ride Transaction Entity**

* Every ride transaction should be identified by a unique transaction id.
* Every ride transaction should have corresponding customer id linked to it.
* Every ride transaction should capture the Start timestamp and pickup station for respective ride.
* Bike ID and Station ID should match to ensure correct bike is picked up from correct Station.
* Ride transaction with no drop station and end time should indicate status as in use else should display Completed.
* Ride transaction having different drop station than pickup station should update Bike at station entity to maintain correct bike and station combination.
* Every ride transaction should capture distance travelled and should be updated against the respective Bike ID.

**Employee Entity**

* Every employee should have an employee id
* Every employee should have a first name

**Maintenance Entity**

* Every time a bike comes into servicing, an appropriate maintenance id should be entered across it
* When a bike comes into maintenance, the date of maintenance should be entered
* Every bike that comes into servicing should have a bike id
* The employee id of the employee who services the bike should be entered
* When maintenance of the bike is completed, the date of completion should be entered
* After servicing of the bike is completed, the servicing cost should be entered

**Customer Feedback Entity**

* Every feedback should have a feedback id
* Every customer should give a rating and the rating can be 1, 2, 3, 4, 5
* Every customer who gives a rating or writes feedback should have a customer id
* Every feedback or rating should have a corresponding transaction id associated with it

**Payment Entity**

* Every payment processed should have a payment id associated with it
* The amount that is paid by the customer should be present
* The date on which payment is made should be present
* The payment is accepted only in the form of credit card and mobile wallet
* Every payment transaction id should have the corresponding ride transaction id associated with it
* If discount is offered, then it should have corresponding discount id

**Discount Entity**

* If the discount is offered, then it should have corresponding discount id
* There should be a start date and end date mentioned (validity) for every discount
* The customer id of the customer who gets a discount should be present

Business Views:

**Location Performance**

* This view will show the business team which zip code/location is performing good or bad. Using this data capture from view, business tram will take decisions.

**Ride Frequency**

* Using this view business team will analyse the ride trend daily, weekly, monthly, quarterly, and yearly.

**Customer loyalty**

* Using this view business team will identify which customer are loyal and are frequently using the service. This information will help business team to serve better.

Data Flow Diagrams:

Diagram

Description automatically generated

***Diagram

Description automatically generated***

***Diagram

Description automatically generated***

***Diagram

Description automatically generated***

Entity Relationship Diagram:

***Diagram

Description automatically generated***

Table Entity and Attributes:

Table 1: Customer

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_owner |

Table holds customer details like customer ID, name, Email address etc.

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| Customer ID | number | Primary key for customer records.  Foreign key to Customer\_Address, Customer\_Feedback, Ride\_transaction. |
| First Name | varchar2 | First name of the customer. Should not be NULL. |
| Last Name | varchar2 | Last name of the customer. |
| Gender | varchar2 | M = Male, F = Female, N = Non-Binary, Not NULL |
| Email ID | varchar2 | Email used to login. Not NULL. Unique |
| Contact | number | Phone Number, Not NULL, unique |
| SSN Number | number | Unique number for each customer. Not NULL |
| Customer Type | varchar2 | Casual or Registered, Not NULL |
| DOB | date | Date of birth. NOT NULL |

Table 2: Customer\_Address

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Table holds address ID, city, state, etc.

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| Address\_ID | number | Primary key for customer address table (Customer\_Address), Not NULL, Unique for each customer |
| Line 1 | varchar2 | Street name, Apartment name, Not NULL |
| Line 2 | varchar2 | Apartment Number |
| City | varchar2 | City name, should match the state mentioned, Not NULL |
| State | varchar2 | State name, Not NULL |
| Zip code | varchar2 | Unique to specific area should go in accordance with the given address, Not NULL |
| Type | varchar2 | Home or Work, Not NULL |
| Customer ID | number | Foreign key to Customer\_Address. Primary key to Customer\_ID. Not NULL |

Table 3: Bike

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Table holds Bike ID, Bike Status, etc.

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| Bike ID | number | Primary key for Bike, foreign key to Bike at station, Maintenance, Ride\_Transaction. Must be unique for each bike. |
| Status | varchar2 | Active or inactive, Not NULL |
| Last service | date | Last service date |
| Commission date | date | Commission date, Not NULL |
| Serviced by | number | Maintenance Personnel Name |
| Bike No. | number | Unique for each bike. Not NULL |
| Total distance | number | Updated as and when required based on ride details. |

Table 4: Station

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Table holds information about Station ID, Station Location & Station Capacity.

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| Station\_ID | number | Primary key for Station. Foreign key to Bike at Station. Not NULL. Unique |
| zip code | varchar | Zip code depending on the station address. Not NULL |
| State | varchar2 | State name depending on the station location. Not NULL |
| City | varchar2 | City name depending on the station location. Not NULL |
| Name | varchar | Station name. Not NULL, Unique |
| Capacity | number | Capacity of the bikes at the station., Not NULL |

Table 5: Bike at station

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Table holds information about Bikes parked at Station by Bike ID, Station ID & Bike - Station ID.

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| Bike\_station\_ID | number | Primary key for Bike at station. Unique. Not NULL |
| Bike ID | number | Foreign key. Primary key for Bike. Not NULL, Unique |
| Station\_ID | number | Foreign key. Primary key for Station., Not NULL |

Table 6: Maintenance

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Table holds information about Maintenance ID, Date of Maintenance, Cost etc.

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| Maintenance ID | number | Primary key for Maintenance. Not NULL, Unique |
| Summary | varchar | Summary describing maintenance nature |
| Date of maintenance | date | Date of maintenance. (DD-MOM-YY) |
| Bike ID | number | Foreign key. Primary key for Bike. not NULL |
| Serviced by | number | Maintenance Personnel name |
| Complete date | date | Date of completion. (DD-MOM-YY) |
| Maintenance cost | number | Cost of maintenance. |

Table 7: Ride\_Transaction

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Table holds information about ride details.

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| Trans\_ID | number | Transaction ID for each ride, Primary for Ride\_Transaction |
| Customer ID | number | Foreign key referencing Customer ID in table Customer. Not NULL |
| Payment ID | number | Foreign key, not null |
| Bike pick up station | number | Foreign key referencing Station ID in table Station. Not NULL |
| Bike drop station | number | Foreign key referencing Station ID in table Station. Not NULL |
| Transaction Date | date | Time of transaction. ("DD-MOM-YY HH:MM: SS" (string) or as YYYYMMDDHHMMSS (numeric)). Not NULL |
| Start time | timestamp | Start time. ("DD-MOM-YY HH:MM: SS" (string) or as YYYYMMDDHHMMSS (numeric)). Not NULL |
| End time | sysdate | End time. ("DD-MOM-YY HH:MM: SS" (string) or as YYYYMMDDHHMMSS (numeric)). Not NULL |
| Bike ID | number | Foreign key referencing Bike ID in Bike at Station. Not NULL. Unique |
| Distance | number | Total ride distance |
| Status | varchar2 | Active, inactive, in use. |

Table 8: Feedback

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Table holds information about customer feedback

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| Feedback ID | number | Primary key for Feedback, Not NULL, unique |
| Review | varchar | Will vary with each customer. |
| Rating | number | (1/2/3/4/5), Not NULL |
| Customer ID | number | Foreign key referencing customer. Not NULL. unique |
| Trans\_ID | number | Foreign key referencing Ride\_Transaction. Not NULL, unique. |

Table 9: Discount

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Table holds information about Discount offerings

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| Discount ID | number | Primary key for Discount. Not NULL, unique |
| Customer ID | number | Foreign key, not null, unique |
| Description | varchar2 | Describes the kind of discount offered. Not NULL, |
| Percentage | number | Percent of discount offered. Not NULL, unique. |
| Start date | date | Date of start. Not NULL, unique. |
| End date | date | Date of end. Not Null, unique. |

Table 10: Payment

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Holds information about payment transaction of customer

|  |  |  |
| --- | --- | --- |
| Column | Date type | Description |
| Payment ID | number | Primary key for Payment. Not Null, unique. |
| Customer ID | number | Foreign key, not null, unique |
| Amount | number | Not null |
| Payment Date | date | Payment date, Not NULL |
| Discount ID | number | Foreign key referencing Discount. Not NULL, unique. |
| Extra fee | number | If limited time exceeds, extra fee is charged. |

Table 11: Employee

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Holds information about payment transaction of customer

|  |  |  |
| --- | --- | --- |
| Column | Date type | Description |
| Employee ID | number | Primary key for Payment. Not Null, unique. |
| First name | varchar2 | First name of the employee, should not be null. |
| Last name | varchar2 | Last name of the employee, should not be null |
| Department | number | What department the employee belongs to. |

Table 11: OTP

|  |  |
| --- | --- |
| Schema: | To be decided |
| Responsible: | DB\_OWNER |

Holds information about payment transaction of customer

|  |  |  |
| --- | --- | --- |
| Column | Date type | Description |
| Customer ID | number | Foreign key, Not Null, unique. |
| OTP | number | Generated unique for each transaction |
| OTP generated time | timestamp | Time the OTP was generated |
| OTP expire time | timestamp | Time when the OTP expires |

Instructions to run the Scripts:

**Step 1:** Run the user creation script

**Step 2:** Login as user: MHATREA (password: BikeShare2022) who is the DB owner.

**Step 3:** Run the object creation script.

***Note:*** *Disconnect any connected user before running the user creation script again.*

Scripts:

<1_User_Creation_Script.SQL>

<2_Object_Creation_Script_2.sql>