DATABASE DESIGN

CS 6360.002

BIKE RENTAL SYSTEM

SRIRAAM RAMAKRISHNAN (sxr163730)

SUNIT MATHEW (sxm167631)

JITHIN PAUL (jxp161130)

CONTENTS

INTRODUCTION

As part of our project, we have designed a Bike rental system. A customer can rent a bike of his/her choice from their desired location. The location from where a customer picks up the bike is known as pick-up location and where the customer drops it off is known as the drop off location. Customers reserve a bike for a certain fixed duration and if the return is delayed, they will be charged a late return fee. Customers are also encouraged to avail the insurance option which is provided, but it is not mandatory to do so. Those who own a membership card would be offered a discount on their final amount.

REQUIREMENTS

- The bike rental company has a wide collection of bikes.
- Each bike belongs to a certain category and a certain pick up/drop off location.
- The price of a bike is determined by its make/model.
- A customer can rent any available bike from any location of their preference.
- Customer should be able to opt/purchase Insurance at the time of booking, but it is not mandatory.
- At the time of booking, all available bikes at the different locations are shown to the customer attempting to make the booking. They can then choose accordingly.
- Customers can apply at most one discount code at the time of booking.
- A customer who holds a membership card will by default be charged only 90% of what other customers would be charged. A discount coupon can be applied on top of this.
- The trip ends when the customer returns the bike and the bill is generated upon return.
- A customer can return the bike at any time, but if they are late, a late return fee will be charged in addition to the rental cost.
- Standard tax rates apply and the final bill cost includes rental charge, late fee if any and the tax.
- A bike becomes available if booked by a customer for the period of booking; it becomes available again from the time of return.
- A booking can be cancelled until 24 hours before the actual pick up.

ENTITIES

BIKE

Contains all the different bikes available in the system. Each bike will have attributes like model, make, category and registration number. This entity will also be used to indicate the availability of a particular bike.

CUSTOMER

Anyone who rents a bike from the system is referred to as a customer. The system will maintain all personal details of the customer such as Name, Address, Contact etc. The system can also distinguish between someone who is a member and someone who is not.

LOCATION

Customers can pick up/drop off their bikes from various available locations. Standard geographical features will be used as attributes to identify a location.

BIKE CATEGORY

Each bike present in the system belongs to certain category. Categories include Sports, Vintage, Racing, Normal etc. The price of a bike is determined by the category to which it belongs.

BILLING

A bill is generated upon return of the bike. Bill indicates the various individual components and the grand total amount that the customer will be charged for the trip.

BOOKING

This entity is used to track each booking/reservation that gets made. Each booking will be associated with a customer, duration, bike involved etc. Insurance and discounts are also tracked if applicable to the booking.

INSURANCE

Insurance entity is used to track the type of insurance availed by the customer if any. Contains attributes such as Coverage, insurance name and the per day cost.

DISCOUNT

Discount entity is used to track discounts that may have been used by a customer during a trip. Contains attributes such as Discount percentage, name, expiry etc.

RELATIONSHIPS

BELONGS TO

Helps determine the category to which a bike belongs. This relationship is between Bike and Bike category.

RESIDES AT

Each bike is present at a certain location at a given point in time and this relationship helps determine that. This is between Bike and Location.

COSTS

This relationship helps tag a booking to a billing. So, this exists between Booking and Billing.

CONTAINS

Each booking may or may not have a discount associated with it. This relationship helps identify that. It exists between Booking and Discount.

INCLUDES

Each booking may or may not have an insurance option associated with it. This relationship helps identify that. It exists between Booking and Insurance.

PICK UP LOCATION

This exists between booking and location which indicates that as part of this booking, the customer picked up the bike from the specified location.

DROP OFF LOCATION

This exists between booking and location which indicates that as part of this booking, the customer dropped off the bike at the specified location.

RENTS

A ternary relationship that exists between Bike, Booking and Location. A customer will rent a bike as part of a certain booking. This relationship helps bring that out.

ASSUMPTIONS

- Every booking is not necessarily associated with billing as some bookings may get cancelled.
- Each bike present in the system will be available in some location.
- Each bill may or may not have a discount code associated with it.
- Each booking may or may not have an insurance associated with it.

FUNCTIONAL DEPENDENCIES

Customer Relation:

- DL_number -> Fname, Mname, Lname, Phone_number, Email_id, Street, City, State, Zipcode, Membership_id, Membership_type
- Zipcode -> State, City

Bike Relation:

- Registration_number -> Model, Make, Model_year, Bike_category_name, Loc_id, Availability
- Model -> Make

Bike_category Relation:

- Category_name -> Cost, Late_fee_per_hour

Location Relation:

- Location_id -> Name, Street, City, State, Zipcode
- Zipcode -> State, City

Booking Relation:

- Booking_id -> From_dt_time, Re_dt_time, Amount, Booking_status, Pickup_loc, Drop_off_loc, Reg_num, DL_num, Ins_code, Act_ret_dt_time, Discount_code

Billing Relation:

- Bill_id -> Bill_date, Bill_status, Discount_amt, Total_amt, Tax_amt, Booking_id, Total_late_fee

Discount Relation:

- Discount_code -> Discount_name, Expiry_date, Discount_percentage
- Discount_name -> Discount_code, Expiry_date, Discount_percentage

Insurance Relation:

- Insurance_code -> Insurance_name, Coverage_type, Cost_per_day
- Insurance_name -> Insurance_code, Coverage_type, Cost_per_day

DEPENDENCIES THAT VIOLATED NORMALIZATION

Transitive dependencies: A -> B and B -> C, where B is non-key attribute.

Customer relation

DL_number -> Zipcode and Zipcode -> State, City

Bike relation

Registration_number -> Model_name

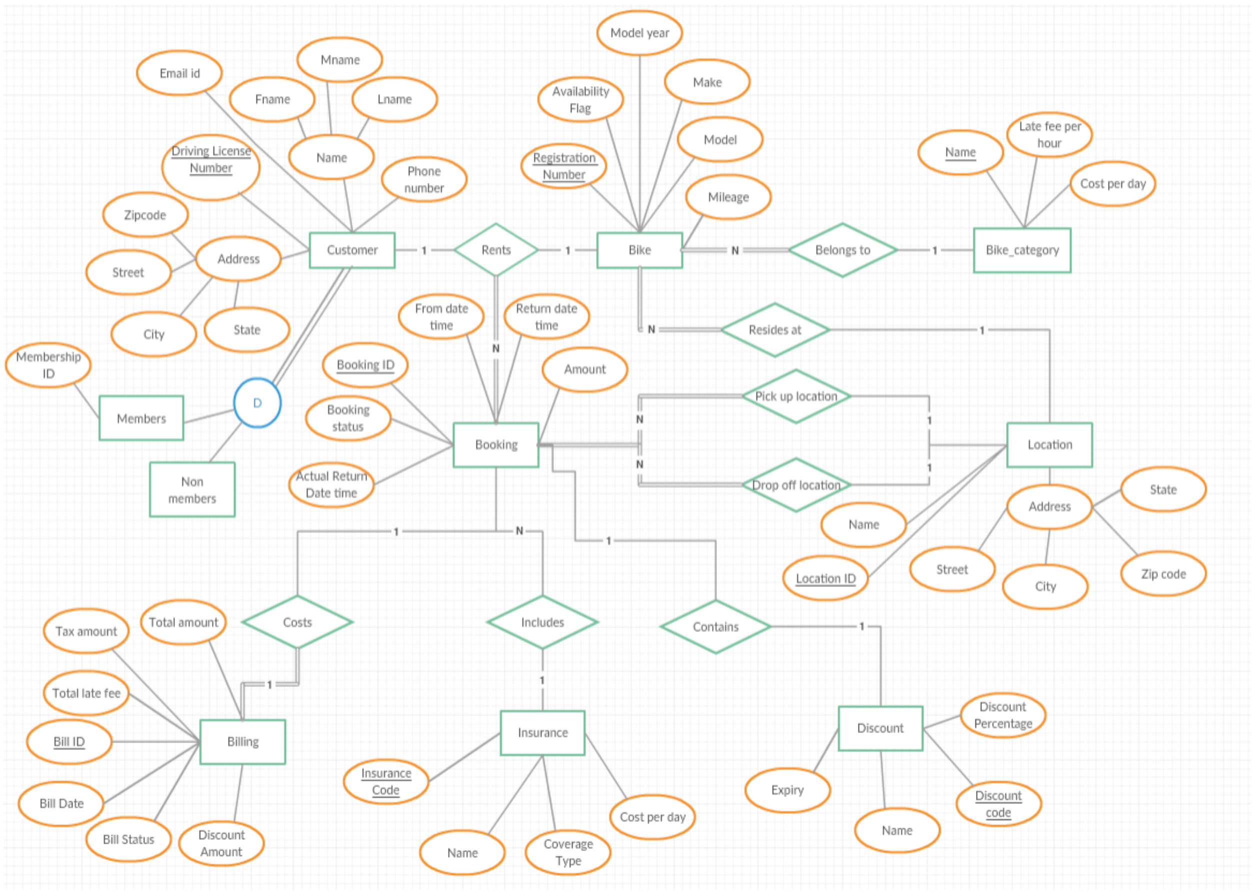
Model_name -> Make

Location relation

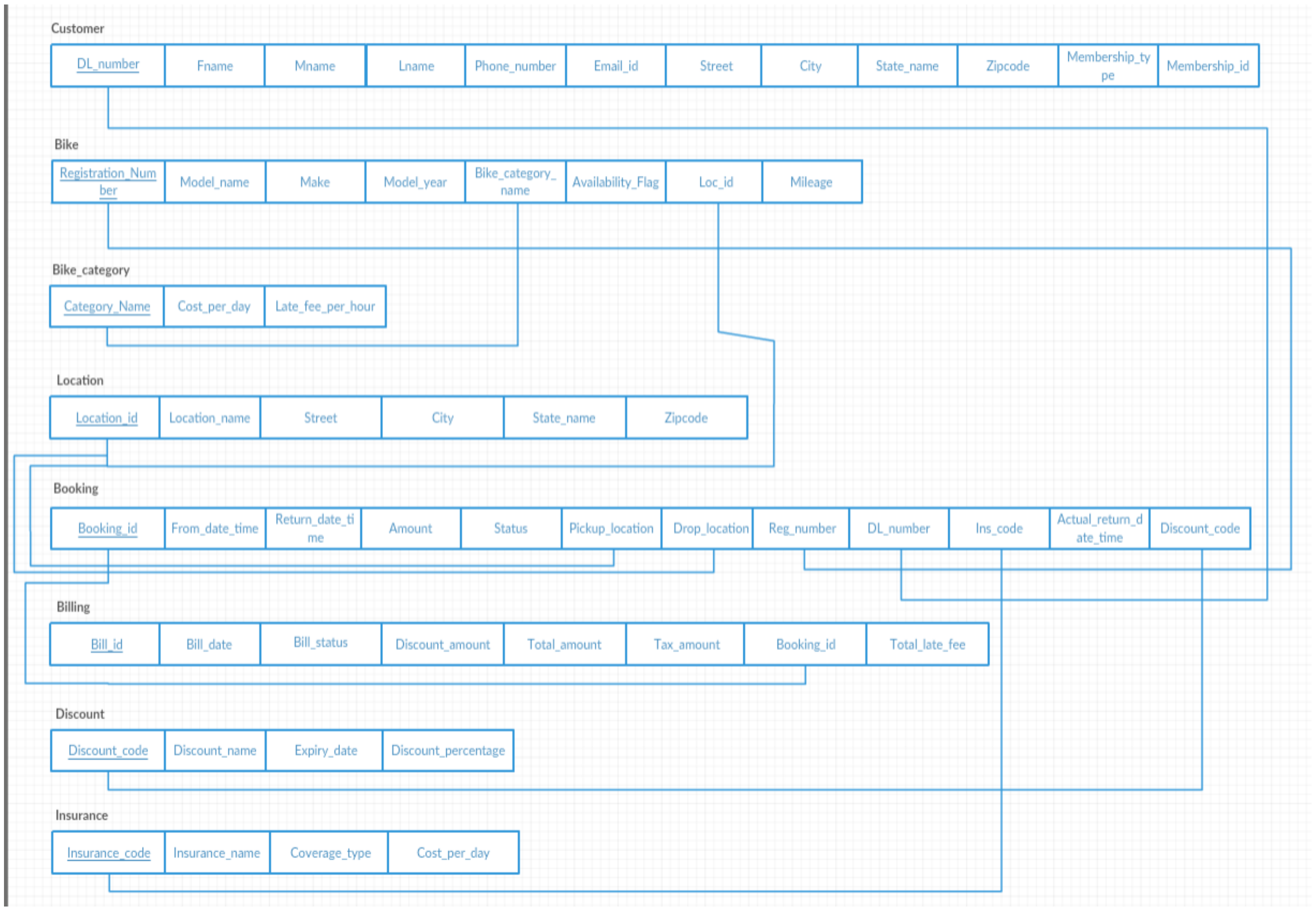
Location_id -> Zipcode

Zipcode -> State, City

ER Diagram


Model year
Email id
Mname
Make
Availability Flag
Fname
Lname
Model
Late fee per hour
Registration Number
Driving License Number
Name
Phone number
Name
Mileage
Cost per day
Zipcode
Customer
1
Rents
1
Bike
N
Belongs to
1
Bike_category
Address
Street
From date time
Return date time
N
Resides at
1
City
State
N
Membership ID
Booking ID
Amount
Pick up location
D
N
1
Members
Booking status
Booking
Location
N
1
Non members
Actual Return Date time
Drop off location
State
Address
Name
1
N
1
Street
Location ID
City
Zip code
Tax amount
Total amount
Costs
Includes
Contains
1
1
1
Total late fee
Discount Percentage
Bill ID
Billing
Insurance
Discount
Insurance Code
Expiry
Discount code
Bill Date
Cost per day
Bill Status
Discount Amount
Name
Coverage Type
Name

RELATIONSHIP SCHEMA (shown in 2NF)


Customer
DL_number | Fname | Mname | Lname | Phone_number | Email_id | Street | City | State_name | Zipcode | Membership_type | Membership_id
Bike
Registration_Number | Model_name | Make | Model_year | Bike_category_name | Availability_Flag | Loc_id | Mileage
Bike_category
Category_Name | Cost_per_day | Late_fee_per_hour
Location
Location_id | Location_name | Street | City | State_name | Zipcode
Booking
Booking_id | From_date_time | Return_date_time | Amount | Status | Pickup_location | Drop_location | Reg_number | DL_number | Ins_code | Actual_return_date_time | Discount_code
Billing
Bill_id | Bill_date | Bill_status | Discount_amount | Total_amount | Tax_amount | Booking_id | Total_late_fee
Discount
Discount_code | Discount_name | Expiry_date | Discount_percentage
Insurance
Insurance_code | Insurance_name | Coverage_type | Cost_per_day

For ease of representation, the above is shown in 2NF. The arrows for the foreign key notations are not shown, but they can be easily understood from the 2 fields that a line connects. For example, Ins_code on **Booking** table is a foreign key that references the Insurance_code on **Insurance** table.

If we normalize the above to 3NF, everything remains the same except the following:

- Customer:
  1. Customer1 (DL_Number, Fname, Mname, Lname, Phone_number, Email_id, Street, Zipcode, Membership_ID, Membership_type)
  2. Customer2 (Zipcode, State_name, City)
- Bike
  1. Bike1 (Registration_number, Model_name, Model_year, Bike_category_name, Availability_flag, Loc_id, Mileage)
  2. Bike2 (Model_name, Make)
- Location
  1. Location1 (Location_Id, Location_name, Street, Zipcode)
  2. Location2 (Zipcode, State_name, City)

CREATE SQL Statements

Customer table

```
CREATE TABLE CUSTOMER

(DL_NUMBER CHAR(8) NOT NULL,

FNAME VARCHAR(25) NOT NULL,

MNAME VARCHAR(15),

LNAME VARCHAR(25) NOT NULL,

PHONE_NUMBER NUMBER(10) NOT NULL,

EMAIL_ID VARCHAR(30) NOT NULL,

STREET VARCHAR(30) NOT NULL,

CITY VARCHAR(30) NOT NULL,

STATE_NAME VARCHAR(20) NOT NULL,

ZIPCODE NUMBER(5) NOT NULL,

MEMBERSHIP_TYPE CHAR(1) DEFAULT 'N' NOT NULL,

MEMBERSHIP_ID CHAR(5),

CONSTRAINT CUSTOMERPK

PRIMARY KEY(DL_NUMBER));
```

Bike_category table

```
CREATE TABLE BIKE_CATEGORY

( CATEGORY_NAME VARCHAR(25) NOT NULL,

COST_PER_DAY NUMBER(5,2) NOT NULL,

LATE_FEE_PER_HOUR NUMBER(5,2) NOT NULL,

CONSTRAINT BIKECATEGORYPK

PRIMARY KEY(CATEGORY_NAME)

);
```

Location table

```
CREATE TABLE LOCATION
( LOCATION_ID CHAR(4) NOT NULL,
LOCATION_NAME VARCHAR(50) NOT NULL,
STREET VARCHAR(30) NOT NULL,
CITY VARCHAR(20) NOT NULL,
STATE_NAME VARCHAR(20) NOT NULL,
ZIPCODE NUMBER(5) NOT NULL,
CONSTRAINT LOCATIONPK
PRIMARY KEY (LOCATION_ID));
```

Bike table

```
CREATE TABLE BIKE
( REGISTRATION_NUMBER CHAR(7) NOT NULL,
MODEL_NAME VARCHAR(25) NOT NULL,
MAKE VARCHAR(25) NOT NULL,
MODEL_YEAR NUMBER(4) NOT NULL,
MILEAGE INTEGER NOT NULL,
BIKE_CATEGORY_NAME VARCHAR(25) NOT NULL,
LOC_ID CHAR(4) NOT NULL,
AVAILABILITY_FLAG CHAR(1) NOT NULL,
CONSTRAINT BIKEPK
PRIMARY KEY (REGISTRATION_NUMBER),
CONSTRAINT BIKEFK1
FOREIGN KEY (BIKE_CATEGORY_NAME) REFERENCES BIKE_CATEGORY(CATEGORY_NAME),
CONSTRAINT BIKEFK2
FOREIGN KEY (LOC_ID) REFERENCES LOCATION (LOCATION_ID)
);
```

Discount table

```
CREATE TABLE DISCOUNT
( DISCOUNT_CODE CHAR(4) NOT NULL,
DISCOUNT_NAME VARCHAR(25) NOT NULL,
EXPIRY_DATE DATE NOT NULL,
DISCOUNT_PERCENTAGE NUMBER(4,2) NOT NULL,
CONSTRAINT DISCOUNTPK
PRIMARY KEY (DISCOUNT_CODE),
CONSTRAINT DISCOUNTSK
UNIQUE (DISCOUNT_NAME)
);
```

Insurance table

```
CREATE TABLE INSURANCE
( INSURANCE_CODE CHAR(4) NOT NULL,
INSURANCE_NAME VARCHAR(50) NOT NULL,
COVERAGE_TYPE VARCHAR(200) NOT NULL,
COST_PER_DAY NUMBER(4,2) NOT NULL,
CONSTRAINT INSURANCEPK
PRIMARY KEY (INSURANCE_CODE),
CONSTRAINT INSURANCESK
UNIQUE (INSURANCE_NAME)
);
```

Booking table

```
CREATE TABLE BOOKING
(BOOKING_ID CHAR(5) NOT NULL,
FROM_DT_TIME TIMESTAMP NOT NULL,
RET_DT_TIME TIMESTAMP NOT NULL,
AMOUNT NUMBER(10,2) NOT NULL,
BOOKING_STATUS CHAR(1) NOT NULL,
PICKUP_LOC CHAR(4) NOT NULL,
DROP_LOC CHAR(4) NOT NULL,
REG_NUM CHAR(7) NOT NULL,
DL_NUM CHAR(8) NOT NULL,
INS_CODE CHAR(4),
ACT_RET_DT_TIME TIMESTAMP,
DISCOUNT_CODE CHAR(4),
CONSTRAINT BOOKINGPK
PRIMARY KEY (BOOKING_ID),
CONSTRAINT BOOKINGFK1
FOREIGN KEY (PICKUP_LOC) REFERENCES LOCATION (LOCATION_ID),
CONSTRAINT BOOKINGFK2
FOREIGN KEY (DROP_LOC) REFERENCES LOCATION (LOCATION_ID),
CONSTRAINT BOOKINGFK3
FOREIGN KEY (REG_NUM) REFERENCES BIKE(REGISTRATION_NUMBER),
CONSTRAINT BOOKINGFK4
FOREIGN KEY (DL_NUM) REFERENCES CUSTOMER (DL_NUMBER),
CONSTRAINT BOOKINGFK5
FOREIGN KEY (INS_CODE) REFERENCES INSURANCE(INSURANCE_CODE),
CONSTRAINT BOOKINGFK6
FOREIGN KEY (DISCOUNT_CODE) REFERENCES DISCOUNT (DISCOUNT_CODE));
```

Billing table

```
CREATE TABLE BILLING
( BILL_ID CHAR(6) NOT NULL,
BILL_DATE DATE NOT NULL,
BILL_STATUS CHAR(1) NOT NULL,
DISCOUNT_AMOUNT NUMBER(10,2) NOT NULL,
TOTAL_AMOUNT NUMBER(10,2) NOT NULL,
TAX_AMOUNT NUMBER(10,2) NOT NULL,
BOOKING_ID CHAR(5) NOT NULL,
TOTAL_LATE_FEE NUMBER(10,2) NOT NULL,
CONSTRAINT BILLINGPK
PRIMARY KEY (BILL_ID),
CONSTRAINT BILLINGFK1
FOREIGN KEY (BOOKING_ID) REFERENCES BOOKING(BOOKING_ID)
);
```

INSERT SQL Statements

Customer table

```
INSERT INTO CUSTOMER VALUES('123456E7', 'GHI',

NULL,'GHI','9098123429', 'ghighi@gmail.com','122 DALLAS DOWNTOWN',

'DALLAS','TEXAS',50932,'N',NULL);

INSERT INTO CUSTOMER VALUES('T0981237', 'DEF',

NULL,'DEF','4245985740', 'defdef@gmail.com','212 MIDWAY STREET',

'RALEIGH','NORTH CAROLINA',42571,'M','M1004');

INSERT INTO CUSTOMER VALUES('F0091266', 'ABC',

NULL,'ABC','7892340918', 'abcabc@gmail.com','300 COLLEGE STATION',

'COLLEGE STATION','TEXAS',56784,'N',NULL);
```

Bike_category table

```
INSERT INTO BIKE_CATEGORY VALUES('OFFROADING',55,30);

INSERT INTO BIKE_CATEGORY VALUES('EXTREME',70,40);

INSERT INTO BIKE_CATEGORY VALUES('RACING',50,30);

INSERT INTO BIKE_CATEGORY VALUES('HYBRID',40,20);

INSERT INTO BIKE_CATEGORY VALUES('NORMAL',30,10);
```

Location table

```
INSERT INTO LOCATION VALUES('L101','DAL AIRPORT',

'Herb Kelleher Way','Dallas','Texas',75235);

INSERT INTO LOCATION VALUES('L102',LA INTL AIRPORT',

'World Way','Los Angeles','California',90045);

INSERT INTO LOCATION VALUES('L103',DFW INTL AIRPORT',

'International Pkwy','DFW Airport','Texas',75261);

INSERT INTO LOCATION VALUES('L104','HOUSTON AIRPORT',
```

'Groschke Rd','Houston','Texas',77094);

Bike table

INSERT INTO BIKE VALUES('KRX1234','ROADRUNNER','KAWASAKI',

2014,100,'RACING','L101','A');

INSERT INTO BIKE VALUES('HCD4567','CRUISER','HONDA',

2015,150,'NORMAL','L102','N');

INSERT INTO BIKE VALUES('HHD3245','HYBRID','HONDA',

2014,123,'HYBRID','L103','A');

INSERT INTO BIKE VALUES('HLD3231','LANDCRUISER','HONDA',

2016,130,'OFFROADING','L102','A');

INSERT INTO BIKE VALUES('YTD2451','EXTRAVAGANZA','YAMAHA',

2015,70,'EXTREME','L103','A');

INSERT INTO BIKE VALUES('HRD3135','RACER','HERO',

2016,110,'RACING','L102','A');

INSERT INTO BIKE VALUES('HTC1355','THUNDERCAT','HARLEY-DAVIDSON',

2016,114,'HYBRID','L101','A');

INSERT INTO BIKE VALUES('HLO2143','LIBERIO','HERO',

2015,123,'NORMAL','L104','A');

Discount table

INSERT INTO DISCOUNT VALUES ('A111','ORACLE',

to_date('2018-01-21','YYYY-MM-DD'),25);

INSERT INTO DISCOUNT VALUES ('B111','AMAZON',

to_date('2019-09-25','YYYY-MM-DD'),20);

INSERT INTO DISCOUNT VALUES ('C111','WEEKENDER SPL',

to_date('2017-10-14','YYYY-MM-DD'),10);

Insurance table

INSERT INTO INSURANCE VALUES('I111', 'COLLISION/ MINOR ACCIDENT WAIVER', 'Covers theft and damage to the rental bike',15);

INSERT INTO INSURANCE VALUES('I112', 'SUPPLEMENTAL LIABILITY PROTECTION', 'Covers damage to others',12);

Booking table

INSERT INTO BOOKING VALUES('B1113',TO_TIMESTAMP('2017-04-10 13:00:00',

'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2017-04-15 13:00:00', 'YYYY-MM-DD HH24:MI:SS'),

450,'R','L101','L101','KRX1234','123456E7','I111',

TO_TIMESTAMP('2017-04-16 13:00:00', 'YYYY-MM-DD HH24:MI:SS'),'A111');

INSERT INTO BOOKING VALUES('B1114',TO_TIMESTAMP('2017-04-23 13:00:00',

'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2017-04-25 20:30:00', 'YYYY-MM-DD HH24:MI:SS'),

48,'R','L102','L102','HCD4567','T0981237','I112',

TO_TIMESTAMP('2017-04-25 20:30:00', 'YYYY-MM-DD HH24:MI:SS'),'B111');

Billing table

INSERT INTO BILLING VALUES('BL1111',to_date('2017-04-16','YYYY-MM-DD'),

'P',35 ,138.03,12.38 ,'B1113',720);

INSERT INTO BILLING VALUES('BL1112',to_date('2017-04-25','YYYY-MM-DD'),

'P',85 ,487.13 ,12.38 ,'B1114',0);

PL/SQL Statements

PROCEDURES

Procedure 1: Calculating discount amount.

Customers who are members are given an additional 10% discount on top of another discount that they may choose to avail.

```
CREATE OR REPLACE PROCEDURE DISCOUNT_AMOUNT

(dlNum IN CUSTOMER.DL_NUMBER%TYPE,

amount IN BILLING.TOTAL_AMOUNT%TYPE,

discountCode IN DISCOUNT.DISCOUNT_CODE%TYPE,

discountAmt OUT BILLING.DISCOUNT_AMOUNT%TYPE) AS

--local declarations

memberType CUSTOMER.MEMBERSHIP_TYPE%TYPE;

discountPercentage DISCOUNT.DISCOUNT_PERCENTAGE%TYPE;

BEGIN

SELECT MEMBERSHIP_TYPE INTO memberType FROM CUSTOMER

WHERE DL_NUMBER = dlNum;

IF NVL(discountCode,'NULL') <> 'NULL' THEN

SELECT DISCOUNT_PERCENTAGE INTO discountPercentage

FROM DISCOUNT WHERE DISCOUNT_CODE = discountCode;

IF memberType = 'M' THEN

discountAmt := amount * ((discountPercentage+10)/100);

ELSE

discountAmt := amount * (discountPercentage/100);

END IF;

ELSE

IF memberType = 'M' THEN
```

```
discountAmt := amount * 0.1;

ELSE

discountAmt := 0;

END IF;

END IF;

END;

/
```

Procedure 2: For calculating late fee and tax.

If the bike is returned after the return time indicated as part of the booking, then a late fee is charged depending the amount of delay and the category to which the bike belongs. The tax on that is also calculated.

```
CREATE OR REPLACE PROCEDURE LATE_FEE_AND_TAX

(actualReturnDateTime IN BOOKING.ACT_RET_DT_TIME%TYPE,

ReturnDateTime IN BOOKING.RET_DT_TIME%TYPE,

regNum IN BOOKING.REG_NUM%TYPE,

amount IN BOOKING.AMOUNT%TYPE,

totalLateFee OUT BILLING.TOTAL_AMOUNT%TYPE,

totalTax OUT BILLING.TAX_AMOUNT%TYPE ) AS

--local declarations

lateFeePerHour BIKE_CATEGORY.LATE_FEE_PER_HOUR%TYPE;

hourDifference DECIMAL(10,2);

BEGIN

SELECT LATE_FEE_PER_HOUR INTO lateFeePerHour

FROM BIKE_CATEGORY BC INNER JOIN BIKE B ON BC.CATEGORY_NAME =

B.BIKE_CATEGORY_NAME WHERE B.REGISTRATION_NUMBER = regNum;

IF actualReturnDateTime > ReturnDateTime THEN

hourDifference := (TO_DATE (TO_CHAR (actualReturnDateTime,
'dd/mm/yyyy hh24:mi:ss'), 'dd/mm/yyyy hh24:mi:ss')
- TO_DATE (TO_CHAR (ReturnDateTime, 'dd/mm/yyyy hh24:mi:ss')
,'dd/mm/yyyy hh24:mi:ss'))*(24);
totalLateFee := hourDifference * lateFeePerHour;
ELSE
totalLateFee := 0;
END IF;
totalTax := (amount + totalLateFee)*0.0825;
END;
/
```

Procedure 3: Calculating overall revenue per month

```
CREATE OR REPLACE PROCEDURE REVENUE_REPORT AS
--local declarations
thisLocationID LOCATION.LOCATION_ID%TYPE;
currentLocationID LOCATION.LOCATION_ID%TYPE;
locationName LOCATION.LOCATION_NAME%TYPE;
thisCategoryName BIKE_CATEGORY.CATEGORY_NAME%TYPE;
thisNoOfBikes integer; thisRevenue DECIMAL(15,2);
--Cursor declaration
CURSOR CURSOR_REPORT IS SELECT TABLE1.LOCATIONID, TABLE1.CATNAME ,
TABLE1.NOOFBIKES,SUM(NVL((TABLE2.AMOUNT),0)) AS REVENUE
FROM (SELECT LC.LID AS LOCATIONID, LC.CNAME AS CATNAME ,
COUNT(B.REGISTRATION_NUMBER) AS NOOFBIKES FROM (SELECT
L.LOCATION_ID AS LID, BC.CATEGORY_NAME AS CNAME FROM
BIKE_CATEGORY BC CROSS JOIN LOCATION L) LC LEFT OUTER JOIN
BIKE B ON LC.CNAME = B.BIKE_CATEGORY_NAME AND LC.LID = B.LOC_ID
```

```
GROUP BY LC.LID, LC.CNAME ORDER BY LC.LID) TABLE1 LEFT OUTER JOIN
(SELECT BBC.PLOC AS PICKLOC,BBC.CNAME AS CNAMES, SUM(BL.TOTAL_AMOUNT) AS
AMOUNT FROM (SELECT B.PICKUP_LOC AS PLOC, B1.BIKE_CATEGORY_NAME AS CNAME,
B.BOOKING_ID AS BID FROM BOOKING B INNER JOIN BIKE B1 ON
B.REG_NUM = B1.REGISTRATION_NUMBER) BBC INNER JOIN BILLING BL
ON BBC.BID = BL.BOOKING_ID WHERE
(to_date (SYSDATE,'dd-MM-yyyy') - to_date(BL.BILL_DATE,'dd-MM-yyyy'))
<=30 GROUP BY BBC.PLOC,BBC.CNAME ORDER BY BBC.PLOC) TABLE2
ON TABLE1.LOCATIONID=TABLE2.PICKLOC AND TABLE1.CATNAME = TABLE2.CNAMES
GROUP BY TABLE1.LOCATIONID, TABLE1.CATNAME, TABLE1.NOOFBIKES
ORDER BY TABLE1.LOCATIONID;
BEGIN
dbms_output.put_line(' ');
dbms_output.put_line('Revenue Report');
OPEN CURSOR_REPORT;
FETCH CURSOR_REPORT INTO thisLocationID, thisCategoryName,
thisNoOfBikes, thisRevenue;
IF CURSOR_REPORT%NOTFOUND THEN
dbms_output.put_line('No Report to be generated');
ELSE
currentLocationID := thisLocationID;
<<LABEL_NEXTLOC>>
SELECT LOCATION_NAME INTO locationName from LOCATION
WHERE LOCATION_ID = currentLocationID;
dbms_output.put_line('Location Name: '|| locationName);
dbms_output.put_line(' ');
dbms_output.put_line('Bike Category' || ' '||'Number of Bikes'
||' '|| 'Revenue');
dbms_output.put_line('------------' || ' '||'--------------'
||' '|| '-------');
dbms_output.put_line(thisCategoryName ||
RPAD(' ', (16 - LENGTH(thisCategoryName)))||thisNoOfBikes
||RPAD(' ', (18 - LENGTH(thisNoOfBikes)))|| thisRevenue);
LOOP
FETCH CURSOR_REPORT INTO thisLocationID, thisCategoryName,
thisNoOfBikes, thisRevenue;
EXIT WHEN (CURSOR_REPORT%NOTFOUND);
IF thisLocationID = currentLocationID THEN
dbms_output.put_line(thisCategoryName ||
RPAD(' ', (16 - LENGTH(thisCategoryName)))||thisNoOfBikes
||RPAD(' ', (18 - LENGTH(thisNoOfBikes)))|| thisRevenue);
ELSE
currentLocationID := thisLocationID;
dbms_output.put_line(' ');
dbms_output.put_line('**********************
*******************************************
***********************************');
dbms_output.put_line(' ');
GOTO LABEL_NEXTLOC;
END IF;
END LOOP;
END IF;
END;
/
```

TRIGGERS

Trigger to generate bill upon completion of trip

This trigger adds a row to the Billing table once the actual date of return of the bike is updated in the booking table and the booking status is updated to R. So, this is triggered after an update on the Booking table.

```
CREATE OR REPLACE TRIGGER BILLING

AFTER UPDATE ON BOOKING

FOR EACH ROW

WHEN (NVL(TO_CHAR(NEW.ACT_RET_DT_TIME),'NULL') <> 'NULL' AND

NEW.BOOKING_STATUS ='R')

DECLARE

-- declaration section

lastBillId BILLING.BILL_ID%TYPE;

newBillId BILLING.BILL_ID%TYPE;

discountAmt BILLING.DISCOUNT_AMOUNT%TYPE;

totalLateFee BILLING.TOTAL_LATE_FEE%TYPE;

totalTax BILLING.TAX_AMOUNT%TYPE;

totalAmountBeforeDiscount BILLING.TOTAL_AMOUNT%TYPE;

finalAmount BILLING.TOTAL_AMOUNT%TYPE;

BEGIN

SELECT BILL_ID INTO lastBillId FROM ( SELECT BILL_ID, ROWNUM AS

RN FROM BILLING)

WHERE RN= (SELECT MAX(ROWNUM) FROM BILLING);

newBillId := 'BL' || TO_CHAR(TO_NUMBER(SUBSTR(lastBillId,3))+1);

CALCULATE_LATE_FEE_AND_TAX(:NEW.ACT_RET_DT_TIME, :NEW.RET_DT_TIME,

:NEW.REG_NUM,:NEW.AMOUNT, totalLateFee, totalTax);

totalAmountBeforeDiscount := :NEW.AMOUNT + totalLateFee + totalTax;

CALCULATE_DISCOUNT_AMOUNT(:NEW.DL_NUM, totalAmountBeforeDiscount,
:NEW.DISCOUNT_CODE, discountAmt);
finalAmount := totalAmountBeforeDiscount - discountAmt;
--insert new bill into the billing_details table
INSERT INTO BILLING (BILL_ID,BILL_DATE,BILL_STATUS,DISCOUNT_AMOUNT,
TOTAL_AMOUNT,TAX_AMOUNT,BOOKING_ID,TOTAL_LATE_FEE)
VALUES (newBillId,to_date(SYSDATE,'YYYY-MM-DD'),'P',
discountAmt,finalAmount,totalTax,:NEW.BOOKING_ID,totalLateFee);
END;
/
```

Trigger to update Bike

This trigger updates the availability flag and location of the bike in the bike table.

```
CREATE OR REPLACE TRIGGER UPDATE_BIKE
AFTER UPDATE ON BOOKING
FOR EACH ROW
WHEN (NVL(TO_CHAR(NEW.ACT_RET_DT_TIME),'NULL') <> 'NULL' OR NEW.BOOKING_STATUS
='C')
DECLARE
BEGIN
IF :NEW.BOOKING_STATUS ='C' THEN
UPDATE BIKE SET AVAILABILITY_FLAG = 'A' , LOC_ID = :NEW.PICKUP_LOC WHERE
REGISTRATION_NUMBER = :NEW.REG_NUM;
ELSE
IF NVL(TO_CHAR(:NEW.ACT_RET_DT_TIME),'NULL') <> 'NULL' THEN
UPDATE BIKE SET AVAILABILITY_FLAG = 'A' , LOC_ID = :NEW.DROP_LOC WHERE
REGISTRATION_NUMBER =
:NEW.REG_NUM;
END IF;
END IF;

END;

/
```

CONCLUDING THOUGHTS

Doing the project has been a great learning experience. We got several insights into the key concepts that go into the creation of a good database system – such as the usage of the ER Diagram, building the Relational Schema and the need for normalization. SQL and PL/SQL helped us realize the different ways in which we could store and manipulate the data.