Database Design - Tour Booking Agency

(CS 6360.004) – Final Project

Anandan Sundar (AXS156730) - Sunny Anand (SXA151231)

INDEX:

§ 1: Introduction

This report outlines the steps involved in designing a database for a tour booking agency and ultimately shows the final design. The database is designed for use by a tour booking system comparable to MakeMyTrip, Expedia. The tour booking agency provides customers to book services such as Round-trip package, Journey and Hotel stay. The customer can have a trip by Bus, Train or Flight. In order to provide these services we must have details about the trains, buses and flights running from a source and destination. Hence, our design must also accommodate record keeping for these entities. Furthermore our design must consider the relationships of these entities in depth to determine how best to store the data. In the following sections we establish the data requirements of the system, develop an ER Diagram from those

data requirements, and then map that ER Diagram into the Relational Schema which, after refinement, can be used to implement the database.

§ 2: Project Objective

- The primary goal of this project is to help the passengers make tour bookings.
- The project is designed to provide customers important and essential functionalities with respect to tour booking.
- The first and important functionality is to provide customers with logistics.
- Secondly, each tour booking should contain information about the accommodation that a customer needs and we have taken care of this as well.
- Thirdly, every transaction that a customer makes is captured and presented in the form of transactions that he has created.
- The system will be able to find transport facilities fulfilling passenger's needs.

§ 3: Project Assumptions

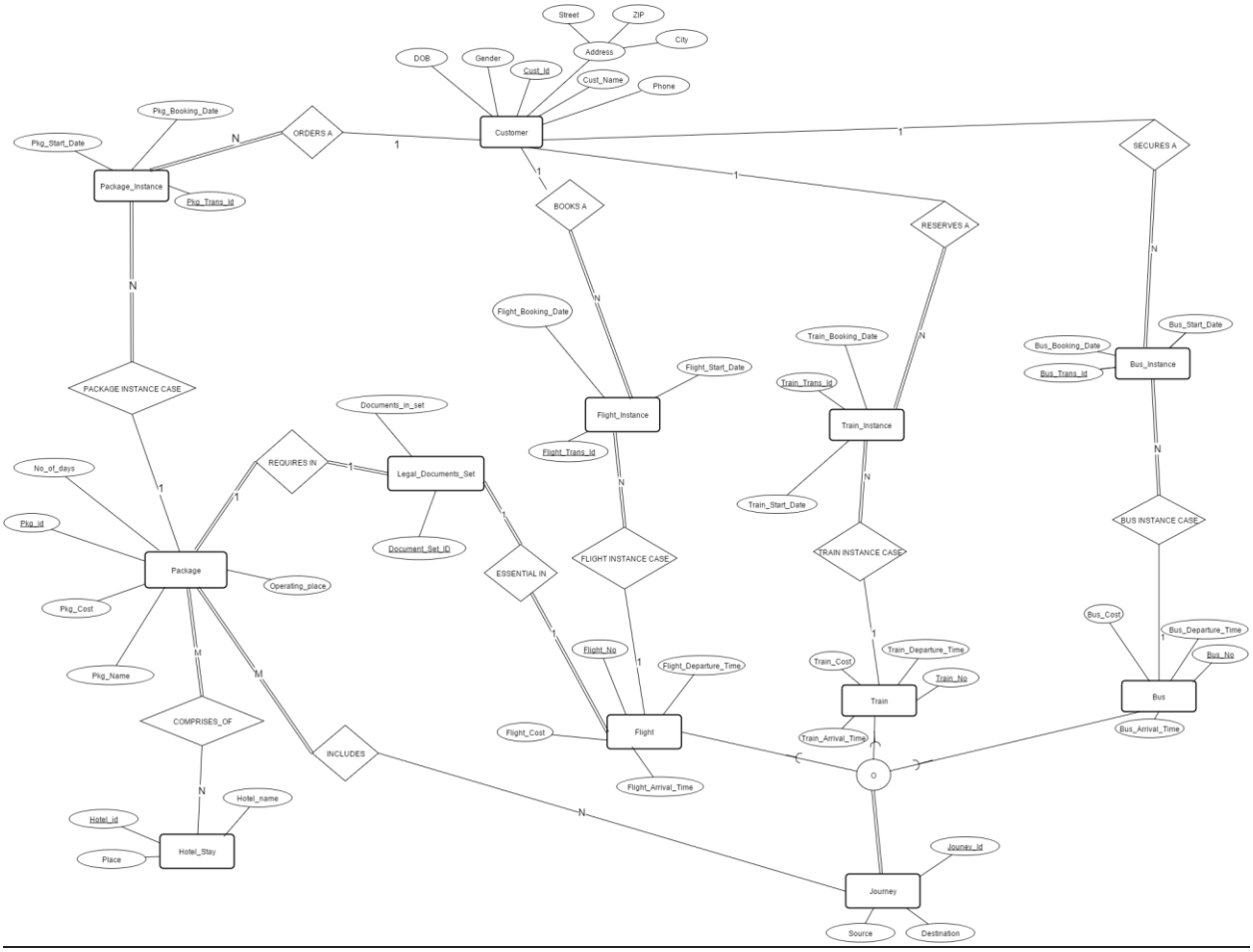
- A customer can book a journey or a round trip package.
- A Journey consists of a source and a destination.
- A journey can be accomplished either by Bus, Train or Flight.
- There might be various ways of transport from a source to destination.
- It's the customer's wish to choose the desired way of travel either by Bus, Train or Flight.
- The system will allow a passenger to reserve a holiday package.
- By booking a package, it will make arrangements for hotel stay to the customer.
- The total expenditure along with discounted price is also available for each customer in a package.
- Overnight stay is also allowed for a user

- All legal documents necessary for a tour are also informed to a customer during a tour booking.
- For International travel in flights, the customer needs legal documents such as Passport, Immigration documents as necessity.
- Our assumption is that all the Bus and Train journeys are domestic.
- On booking a round-trip package, all packages will have one overnight stay at the minimum.
- The overnight stay is accomplished by hotel.
- The ticket cost of the package is inclusive of the journey and the stay cost.

§ 4: ER Diagram

Our ER Diagram shows has entities as `Customer, Legal_document_set, Journey, Package, Hotel_stay, Flight, Bus, Train, Flight_instance, Bus_instance, Train_instance, Package_instance`. Explaining more on, starting with the `Journey,` a journey can be accomplished by either Bus, Train or Flight. So, understandably, `Journey` is a super class of `Flight, Bus, Train.` The fact is, for a same journey, it can be accomplished by any of Flight, Bus and Train. So it is an overlap. Since all journeys are to be implemented by either of the three ways, there exists a total participation. Since every package consists of a journey, there exists a relationship between the `Journey and Package` as Journey includes Package. Since all packages include journey, there will exist a total participation between Package and Includes, but the participation between the Journey and includes is not total since not all Journeys are Packages. The Cardinality ratio here is M:N meaning, a journey can be in multiple packages and a package can have multiple journeys.

Now, coming in terms of the `Customer.` The customer is related with the `Flight_instance, Bus_instance, Train_instance, Package_instance.` That is, the customer can book a trip. He will have a unique transaction ID associated with every booking that he has made. A customer can book N flights but a particular flight instance can only booked by 1 customer. So the cardinality here is 1:N. The participation between flight instance and booking will be a total participation, whereas the participation with respect to the customer will be partial. Same idea will apply to `Bus_instance and Train_instance` as well but the design assumes that all buses, trains operate only domestically. So naturally, there exists a `Legal_document_set` wherein there will be a set of documents which are required for international flights (Passport and Immigration documents). So, domestic flights should have documents as any National ID. Since all flights require some kind of documents, the participation on both sides will be total. Now, coming to `Packages,` a package can have multiple journeys associated with it as previously stated. If a customer books a package, The tour booking agency will associate the package with `Hotel_stay`. The Hotel_stay will provide accommodation to the customer appropriately in between journeys while accomplishing a package. A package will have multiple stays and a stay might accomplish in multiple packages. So the cardinality ratio here is M:N. You can relate these functionalities in the below EER diagram.



1-1 Relationships:

```
Legal_Document_Set —1— <Requires in> —1— Package

Legal_Document_Set —1— <Essential in> —1— Flight
```

1-N Relationships:

```
Customer —1— <Orders_a> —N— Package_Instance

Customer —1— <Secures_a> —N— Bus_Instance

Customer —1— <Reserves_a> —N— Train_Instance

Customer —1— <Books_a> —N— Flight_Instance

Package —1— <Package_instance_case> —N— Package_Instance

Bus —1— < Bus_instance_case> —N— Bus_Instance

Train —1— <Train_instance_case> —N— Train_Instance
```

Flight —1— <Flight_instance_case> —N— Flight_Instance

M-N Relationships:

Package —M— <Comprises_of> —N— Hotel_stay

Package —M— <Includes> —N— Journey

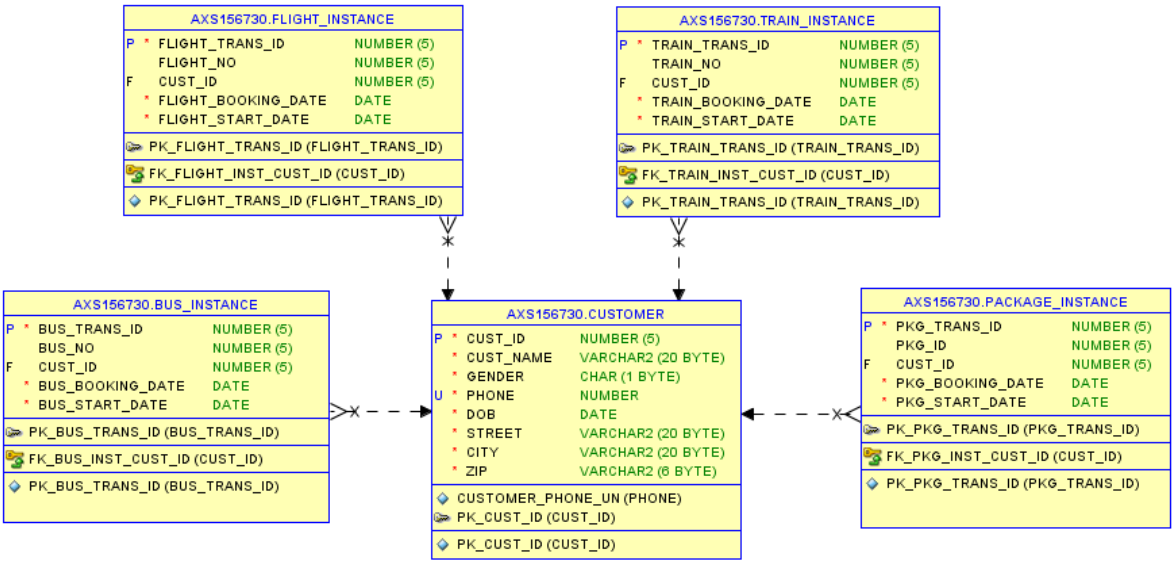
§ 5: Modules in the project

1. Customer Module

a. This module maintains a detailed information for each user.

b. It keeps a track of each transaction the user has made or is going to make in future.

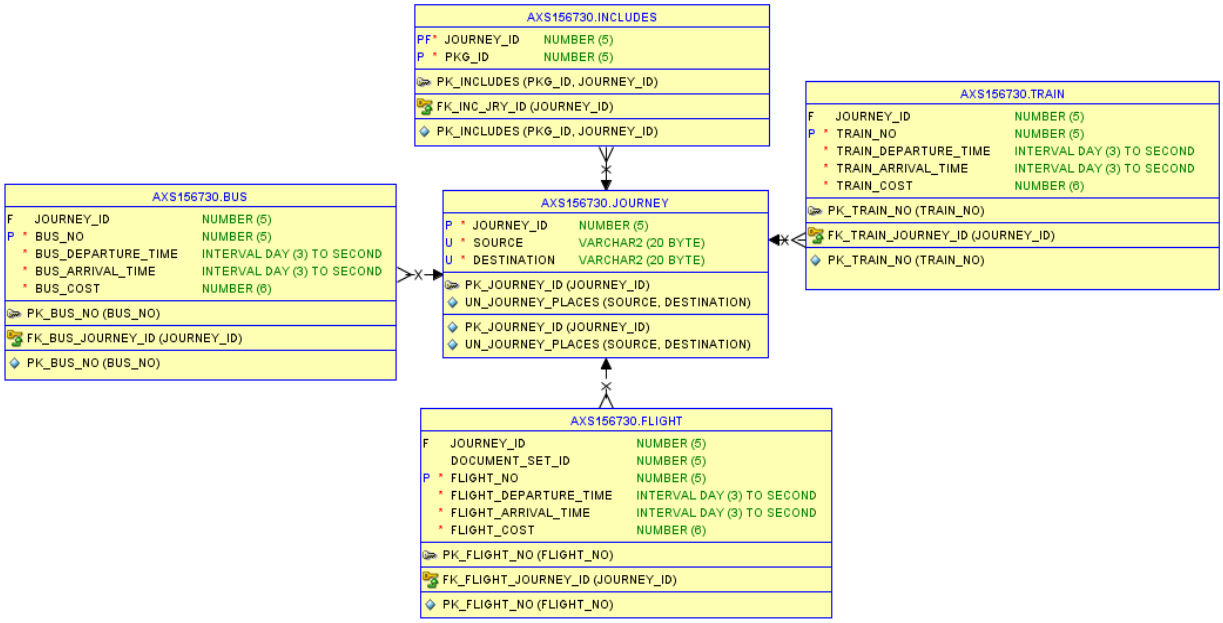
c. It also provides document details the user needs to accomplish a tour.



2. Journey Module

a. This module comes with the functionality of allowing a user to choose for the same tour from different options of bus, train and flights.

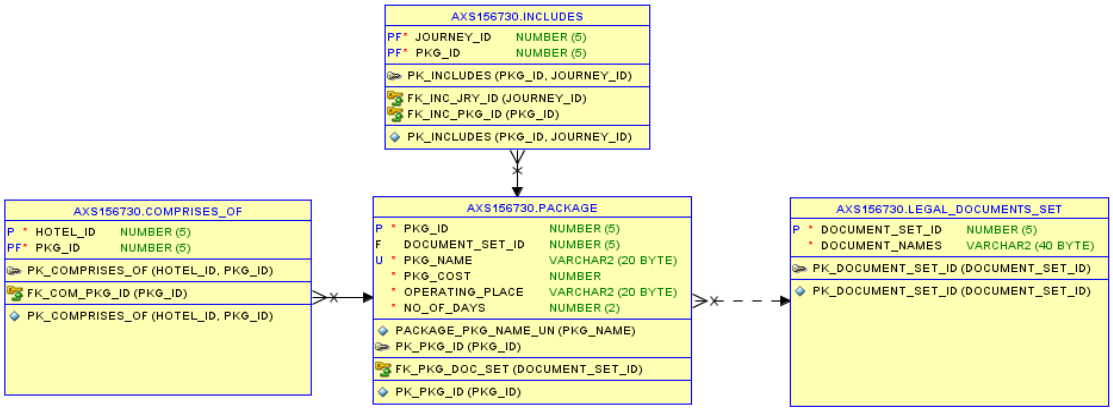
b. The module allows for creating independent travel instances independent of Package module when a customer only wants to use the service of travel.



3. Package Module

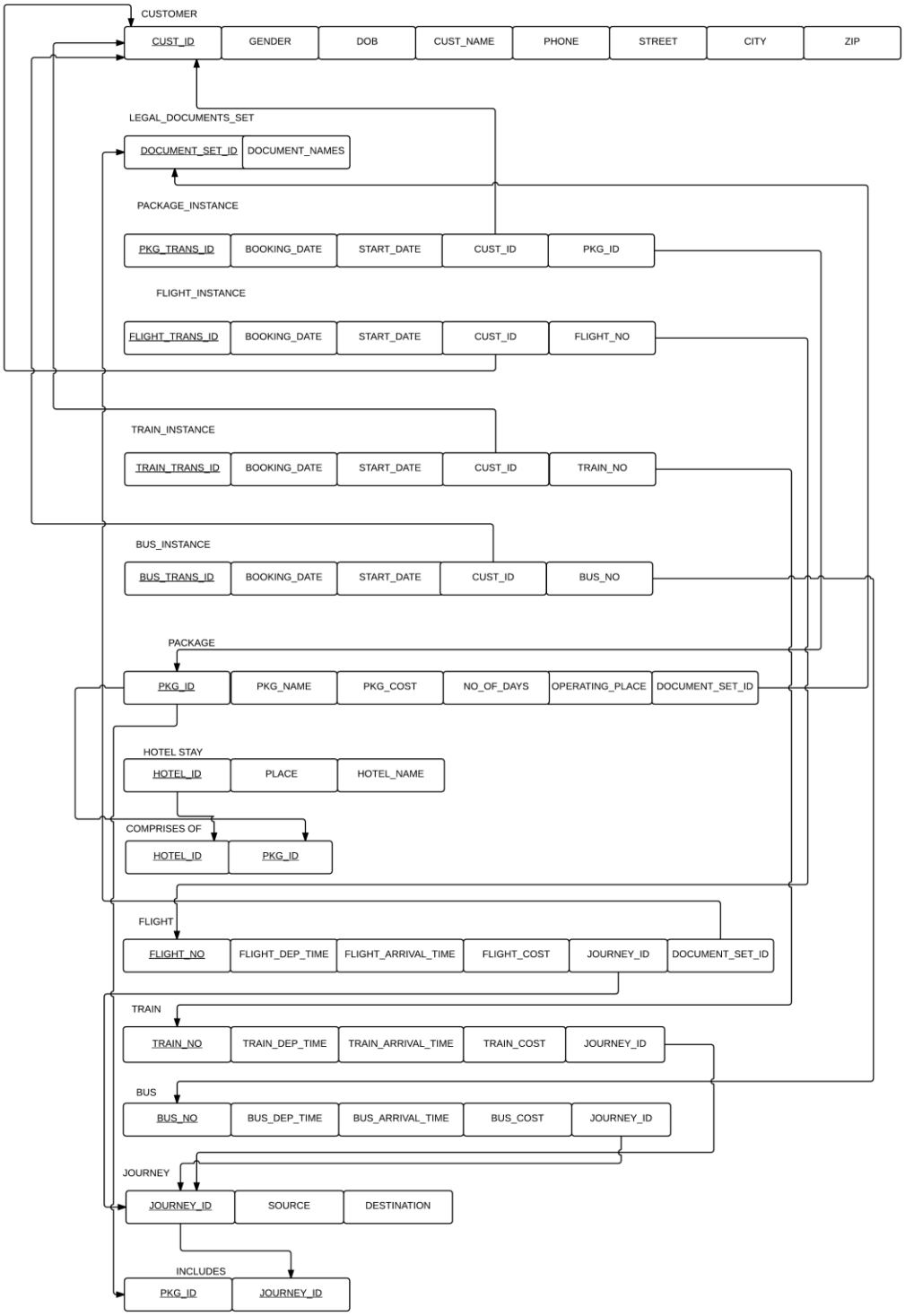
a. This gives the user the flexibility to choose any package with a discount price for a source destination combination.

b. Each package involves booking a hotel for overnight stay so that each customer can take a package and not be worried of the accommodation.



§ 6: Mapping to Relational Schema

We start by mapping the `Customer` entity. We create a relation `Customer` and include all the simple attributes. For the `Address` we include `Street`, `City`, `Zip`, and `State` since it's a composite attribute. We simply create all relations for all entities in that EER. For obtaining attributes, in 1:N relation, the foreign key is made on the N side, for 1:1 relation, the forign key can be made in either of one sides and for M:N relation, there will be a new table created, which has both the foreign keys of entities. We have 7 1:N relations, 2 1:1 relations and 2 M:N relations. The tables `Comprises_of, Includes` are formed by mapping M:N relations. In converting specializations and generalizations we use rule 8A which states that the primary key of the super class is the foreign key of all of its subclasses. We implement this in the `Journey, Train, Bus, Flight` structures. Our relational schema is given below.


CUSTOMER
CUST_ID
GENDER
DOB
CUST_NAME
PHONE
STREET
CITY
ZIP
LEGAL_DOCUMENTS_SET
DOCUMENT_SET_ID
DOCUMENT_NAMES
PACKAGE_INSTANCE
PKG_TRANS_ID
BOOKING_DATE
START_DATE
CUST_ID
PKG_ID
FLIGHT_INSTANCE
FLIGHT_TRANS_ID
BOOKING_DATE
START_DATE
CUST_ID
FLIGHT_NO
TRAIN_INSTANCE
TRAIN_TRANS_ID
BOOKING_DATE
START_DATE
CUST_ID
TRAIN_NO
BUS_INSTANCE
BUS_TRANS_ID
BOOKING_DATE
START_DATE
CUST_ID
BUS_NO
PACKAGE
PKG_ID
PKG_NAME
PKG_COST
NO_OF_DAYS
OPERATING_PLACE
DOCUMENT_SET_ID
HOTEL STAY
HOTEL_ID
PLACE
HOTEL_NAME
COMPRISES OF
HOTEL_ID
PKG_ID
FLIGHT
FLIGHT_NO
FLIGHT_DEP_TIME
FLIGHT_ARRIVAL_TIME
FLIGHT_COST
JOURNEY_ID
DOCUMENT_SET_ID
TRAIN
TRAIN_NO
TRAIN_DEP_TIME
TRAIN_ARRIVAL_TIME
TRAIN_COST
JOURNEY_ID
BUS
BUS_NO
BUS_DEP_TIME
BUS_ARRIVAL_TIME
BUS_COST
JOURNEY_ID
JOURNEY
JOURNEY_ID
SOURCE
DESTINATION
INCLUDES
PKG_ID
JOURNEY_ID

§ 7: Functional Dependencies

Below are the Functional Dependencies for each table in the Relational Schema:

CUSTOMER = {CUST_ID, CUST_NAME, GENDER, STREET, CITY, ZIP, PHONE, DOB}

FD1:

CUST_ID -> {CUST_NAME, GENDER, STREET, CITY, ZIP, PHONE, DOB}

JOURNEY = {JOURNEY_ID, SOURCE, DESTINATION}

FD1: JOURNEY_ID -> {SOURCE, DESTINATION}

LEGAL_DOCUMENTS_SET = {DOCUMENT_SET_ID, DOCUMENT_NAMES}

FD1: DOCUMENT_SET_ID -> {DOCUMENT_NAMES}

PACKAGE = {PKG_ID, DOCUMENT_SET_ID, PKG_NAME, PKG_COST, OPERATING_PLACE, NO_OF_DAYS}

FD1: PKG_ID -> {DOCUMENT_SET_ID, PKG_NAME, PKG_COST, OPERATING_PLACE, NO_OF_DAYS}

HOTEL_STAY = {HOTEL_ID, HOTEL_NAME, PLACE}

FD1: HOTEL_ID -> {HOTEL_NAME, PLACE}

COMPRISES_OF = {HOTEL_ID, PKG_ID}

INCLUDES = {JOURNEY_ID, PACKAGE_ID}

FLIGHT = {FLIGHT_NO, JOURNEY_ID, DOCUMENT_SET_ID, FLIGHT_DEPARTURE_TIME, FLIGHT_ARRIVAL_TIME, FLIGHT_COST}

FD1: {FLIGHT_NO} -> {JOURNEY_ID, DOCUMENT_SET_ID,
FLIGHT_DEPARTURE_TIME, FLIGHT_ARRIVAL_TIME, FLIGHT_COST}

**TRAIN = {TRAIN_NO, JOURNEY_ID, TRAIN_DEPARTURE_TIME,
TRAIN_ARRIVAL_TIME, TRAIN_COST}**

FD1: {TRAIN_NO} -> {JOURNEY_ID, TRAIN_DEPARTURE_TIME,
TRAIN_ARRIVAL_TIME, TRAIN_COST}

**BUS = {BUS_NO, JOURNEY_ID, BUS_DEPARTURE_TIME, BUS_ARRIVAL_TIME,
BUS_COST}**

FD1: {BUS_NO} -> {JOURNEY_ID, BUS_DEPARTURE_TIME, BUS_ARRIVAL_TIME,
BUS_COST}

**BUS_INSTANCE = {BUS_TRANS_ID, BUS_NO, CUST_ID, BUS_BOOKING_DATE,
BUS_START_DATE}**

FD1: {BUS_TRANS_ID} -> {BUS_NO, CUST_ID, BUS_BOOKING_DATE,
BUS_START_DATE}

**FLIGHT_INSTANCE = {FLIGHT_TRANS_ID, FLIGHT_NO, CUST_ID,
FLIGHT_BOOKING_DATE, FLIGHT_START_DATE}**

FD1: {FLIGHT_TRANS_ID} -> {FLIGHT_NO, CUST_ID, FLIGHT_BOOKING_DATE,
FLIGHT_START_DATE}

**TRAIN_INSTANCE = {TRAIN_TRANS_ID, TRAIN_NO, CUST_ID,
TRAIN_BOOKING_DATE, TRAIN_START_DATE}**

FD1: {TRAIN_TRANS_ID} -> {TRAIN_NO, CUST_ID, TRAIN_BOOKING_DATE, TRAIN_START_DATE}

PACKAGE_INSTANCE = {PKG_TRANS_ID, PKG_NO, CUST_ID, PKG_BOOKING_DATE, PKG_START_DATE}

FD1: {PKG_TRANS_ID} -> {PKG_NO, CUST_ID, PKG_BOOKING_DATE, PKG_START_DATE}

§ 8: Normalization to 3NF

We had to normalize the composite attributes from the EER.

This relational schema which is derived from the Enhanced Entity Relationship model is already in 3NF. (See above functional dependencies)

§ 9: SQL

Below are the SQL Create statements and the necessary constraints for them.

```
CREATE TABLE CUSTOMER
(CUST_ID NUMBER(5),
CUST_NAME VARCHAR(20) NOT NULL,
GENDER CHAR(1) NOT NULL,
PHONE NUMBER UNIQUE NOT NULL,
DOB DATE NOT NULL,
STREET VARCHAR(20) NOT NULL,
CITY VARCHAR(20)NOT NULL,
ZIP VARCHAR(6) NOT NULL);
ALTER TABLE CUSTOMER ADD CONSTRAINT PK_CUST_ID PRIMARY KEY (CUST_ID);
ALTER TABLE CUSTOMER ADD CONSTRAINT CHK_CUST_PHONE CHECK (LENGTH(PHONE) = 10);
ALTER TABLE CUSTOMER ADD CONSTRAINT CHK_CUST_GENDER CHECK (GENDER = 'M' OR GENDER =
'F');


CREATE TABLE JOURNEY(
JOURNEY_ID NUMBER(5),
SOURCE VARCHAR(20) NOT NULL,
DESTINATION VARCHAR(20) NOT NULL);
ALTER TABLE JOURNEY ADD CONSTRAINT PK_JOURNEY_ID PRIMARY KEY (JOURNEY_ID);
ALTER TABLE JOURNEY ADD CONSTRAINT UN_JOURNEY_PLACES UNIQUE (SOURCE,DESTINATION);
ALTER TABLE JOURNEY ADD CONSTRAINT CK_JOURNEY_PLACES CHECK (SOURCE != DESTINATION);


CREATE TABLE LEGAL_DOCUMENTS_SET(
```

```
DOCUMENT_SET_ID NUMBER(5),
DOCUMENT_NAMES VARCHAR(40) NOT NULL);
ALTER TABLE LEGAL_DOCUMENTS_SET ADD CONSTRAINT PK_DOCUMENT_SET_ID PRIMARY KEY
(DOCUMENT_SET_ID);


CREATE TABLE PACKAGE(
PKG_ID NUMBER(5),
DOCUMENT_SET_ID NUMBER(5),
PKG_NAME VARCHAR(20) UNIQUE NOT NULL,
PKG_COST NUMBER NOT NULL,
OPERATING_PLACE VARCHAR (20) NOT NULL,
NO_OF_DAYS NUMBER(2) NOT NULL);
ALTER TABLE PACKAGE ADD CONSTRAINT PK_PKG_ID PRIMARY KEY (PKG_ID);
ALTER TABLE PACKAGE ADD CONSTRAINT FK_PKG_DOC_SET FOREIGN KEY(DOCUMENT_SET_ID)
REFERENCES LEGAL_DOCUMENTS_SET(DOCUMENT_SET_ID) ON DELETE CASCADE;


CREATE TABLE HOTEL_STAY(
HOTEL_ID NUMBER(5),
HOTEL_NAME VARCHAR(20) UNIQUE NOT NULL,
PLACE VARCHAR(20) NOT NULL);
ALTER TABLE HOTEL_STAY ADD CONSTRAINT PK_HOTEL_ID PRIMARY KEY (HOTEL_ID);


CREATE TABLE COMPRISES_OF(
HOTEL_ID NUMBER(5),
PKG_ID NUMBER(5));
ALTER TABLE COMPRISES_OF ADD CONSTRAINT PK_COMPRISES_OF PRIMARY KEY (HOTEL_ID,PKG_ID);
ALTER TABLE COMPRISES_OF ADD CONSTRAINT FK_COM_HOTEL_ID FOREIGN KEY (HOTEL_ID)
REFERENCES HOTEL_STAY(HOTEL_ID) ON DELETE CASCADE;
ALTER TABLE COMPRISES_OF ADD CONSTRAINT FK_COM_PKG_ID FOREIGN KEY (PKG_ID) REFERENCES
PACKAGE(PKG_ID) ON DELETE CASCADE;


CREATE TABLE INCLUDES(
JOURNEY_ID NUMBER(5),
PKG_ID NUMBER(5));
ALTER TABLE INCLUDES ADD CONSTRAINT PK_INCLUDES PRIMARY KEY (PKG_ID,JOURNEY_ID);
ALTER TABLE INCLUDES ADD CONSTRAINT FK_INC_JRY_ID FOREIGN KEY (JOURNEY_ID) REFERENCES
JOURNEY(JOURNEY_ID) ON DELETE CASCADE;
ALTER TABLE INCLUDES ADD CONSTRAINT FK_INC_PKG_ID FOREIGN KEY (PKG_ID) REFERENCES
PACKAGE(PKG_ID) ON DELETE CASCADE;


CREATE TABLE FLIGHT(
JOURNEY_ID NUMBER(5),
DOCUMENT_SET_ID NUMBER(5),
FLIGHT_NO NUMBER(5),
FLIGHT_DEPARTURE_TIME INTERVAL DAY(3) TO SECOND NOT NULL,
FLIGHT_ARRIVAL_TIME INTERVAL DAY(3) TO SECOND NOT NULL,
FLIGHT_COST NUMBER(6) NOT NULL);
ALTER TABLE FLIGHT ADD CONSTRAINT PK_FLIGHT_NO PRIMARY KEY (FLIGHT_NO);
ALTER TABLE FLIGHT ADD CONSTRAINT FK_FLIGHT_JOURNEY_ID FOREIGN KEY(JOURNEY_ID)
REFERENCES JOURNEY(JOURNEY_ID) ON DELETE CASCADE;
ALTER TABLE FLIGHT ADD CONSTRAINT FK_FLIGHT_DOC_SET FOREIGN KEY(DOCUMENT_SET_ID)
REFERENCES LEGAL_DOCUMENTS_SET(DOCUMENT_SET_ID) ON DELETE CASCADE;


CREATE TABLE BUS(
JOURNEY_ID NUMBER(5),
BUS_NO NUMBER(5),
BUS_DEPARTURE_TIME INTERVAL DAY(3) TO SECOND NOT NULL,
```

```
BUS_ARRIVAL_TIME INTERVAL DAY(3) TO SECOND NOT NULL,
BUS_COST NUMBER(6) NOT NULL);
ALTER TABLE BUS ADD CONSTRAINT PK_BUS_NO PRIMARY KEY (BUS_NO);
ALTER TABLE BUS ADD CONSTRAINT FK_BUS_JOURNEY_ID FOREIGN KEY(JOURNEY_ID) REFERENCES
JOURNEY(JOURNEY_ID) ON DELETE CASCADE;


CREATE TABLE TRAIN(
JOURNEY_ID NUMBER(5),
TRAIN_NO NUMBER(5),
TRAIN_DEPARTURE_TIME INTERVAL DAY(3) TO SECOND NOT NULL,
TRAIN_ARRIVAL_TIME INTERVAL DAY(3) TO SECOND NOT NULL,
TRAIN_COST NUMBER(6) NOT NULL,
NO_OF_HOURS NUMBER(2) NOT NULL);
ALTER TABLE TRAIN ADD CONSTRAINT PK_TRAIN_NO PRIMARY KEY (TRAIN_NO);
ALTER TABLE TRAIN ADD CONSTRAINT FK_TRAIN_JOURNEY_ID FOREIGN KEY(JOURNEY_ID)
REFERENCES JOURNEY(JOURNEY_ID) ON DELETE CASCADE;


CREATE TABLE TRAIN_INSTANCE(
TRAIN_TRANS_ID NUMBER(5),
TRAIN_NO NUMBER(5),
CUST_ID NUMBER(5),
TRAIN_BOOKING_DATE DATE NOT NULL,
TRAIN_START_DATE DATE NOT NULL);
ALTER TABLE TRAIN_INSTANCE ADD CONSTRAINT PK_TRAIN_TRANS_ID PRIMARY KEY
(TRAIN_TRANS_ID);
ALTER TABLE TRAIN_INSTANCE ADD CONSTRAINT FK_TRAIN_INST_TRAIN_NO FOREIGN KEY(TRAIN_NO)
REFERENCES TRAIN(TRAIN_NO) ON DELETE CASCADE;
ALTER TABLE TRAIN_INSTANCE ADD CONSTRAINT FK_TRAIN_INST_CUST_ID FOREIGN KEY(CUST_ID)
REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE;
ALTER TABLE TRAIN_INSTANCE ADD CONSTRAINT CHK_TRAIN_START_DATE CHECK (TRAIN_START_DATE
>= TRUNC(TRAIN_BOOKING_DATE));


CREATE TABLE BUS_INSTANCE(
BUS_TRANS_ID NUMBER(5),
BUS_NO NUMBER(5),
CUST_ID NUMBER(5),
BUS_BOOKING_DATE DATE NOT NULL,
BUS_START_DATE DATE NOT NULL);
ALTER TABLE BUS_INSTANCE ADD CONSTRAINT PK_BUS_TRANS_ID PRIMARY KEY (BUS_TRANS_ID);
ALTER TABLE BUS_INSTANCE ADD CONSTRAINT FK_BUS_INST_BUS_NO FOREIGN KEY(BUS_NO)
REFERENCES BUS(BUS_NO) ON DELETE CASCADE;
ALTER TABLE BUS_INSTANCE ADD CONSTRAINT FK_BUS_INST_CUST_ID FOREIGN KEY(CUST_ID)
REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE;
ALTER TABLE BUS_INSTANCE ADD CONSTRAINT CHK_BUS_START_DATE CHECK (BUS_START_DATE >=
TRUNC(BUS_BOOKING_DATE));


CREATE TABLE FLIGHT_INSTANCE(
FLIGHT_TRANS_ID NUMBER(5),
FLIGHT_NO NUMBER(5),
CUST_ID NUMBER(5),
FLIGHT_BOOKING_DATE DATE NOT NULL,
FLIGHT_START_DATE DATE NOT NULL);
ALTER TABLE FLIGHT_INSTANCE ADD CONSTRAINT PK_FLIGHT_TRANS_ID PRIMARY KEY
(FLIGHT_TRANS_ID);
ALTER TABLE FLIGHT_INSTANCE ADD CONSTRAINT FK_FLIGHT_INST_FLIGHT_NO FOREIGN
KEY(FLIGHT_NO) REFERENCES FLIGHT(FLIGHT_NO) ON DELETE CASCADE;
ALTER TABLE FLIGHT_INSTANCE ADD CONSTRAINT FK_FLIGHT_INST_CUST_ID FOREIGN KEY(CUST_ID)
REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE;
```

```
ALTER TABLE FLIGHT_INSTANCE ADD CONSTRAINT CHK_FLIGHT_START_DATE CHECK
(FLIGHT_START_DATE >= TRUNC(FLIGHT_BOOKING_DATE));


CREATE TABLE PACKAGE_INSTANCE(
PKG_TRANS_ID NUMBER(5),
PKG_ID NUMBER(5),
CUST_ID NUMBER(5),
PKG_BOOKING_DATE DATE NOT NULL,
PKG_START_DATE DATE NOT NULL);
ALTER TABLE PACKAGE_INSTANCE ADD CONSTRAINT PK_PKG_TRANS_ID PRIMARY KEY
(PKG_TRANS_ID);
ALTER TABLE PACKAGE_INSTANCE ADD CONSTRAINT FK_PKG_INST_PKG_NO FOREIGN KEY(PKG_ID)
REFERENCES PACKAGE(PKG_ID) ON DELETE CASCADE;
ALTER TABLE PACKAGE_INSTANCE ADD CONSTRAINT FK_PKG_INST_CUST_ID FOREIGN KEY(CUST_ID)
REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE;
ALTER TABLE PACKAGE_INSTANCE ADD CONSTRAINT CHK_PKG_START_DATE CHECK (PKG_START_DATE
>= TRUNC(PKG_BOOKING_DATE));
```

And below are the SQL Insert statements to show how the database can be populated:

CUSTOMER

```
INSERT INTO CUSTOMER VALUES(1,'RAM','M',9876789567,'20-OCT-1990','2ND STREET','SAN
FRANCISCO','64564');
INSERT INTO CUSTOMER VALUES(2,'KUMAR','M',9876789569,'03-NOV-1991','3RD
STREET','SACREMENTO','64565');
INSERT INTO CUSTOMER VALUES(3,'ALEX','M',9876789560,'23-DEC-1980','1ST
STREET','DALLAS','54864');
INSERT INTO CUSTOMER VALUES(4,'CHARLES','M',9876789517,'10-FEB-1992','2ND
STREET','CHICAGO','75175');
INSERT INTO CUSTOMER VALUES(5,'MONICCA','F',8876789567,'09-MAR-1970','9TH STREET','NEW
YORK','67453');
INSERT INTO CUSTOMER VALUES(6,'SITA','F',9876788567,'08-MAY-1985','6TH
STREET','MIAMI','86735');
INSERT INTO CUSTOMER VALUES(7,'KRISHNA','M',9876689567,'27-OCT-1967','10TH
STREET','LAS VEGAS','52436');
```

JOURNEY

```
INSERT INTO JOURNEY VALUES(1,'DALLAS','HOUSTON');
INSERT INTO JOURNEY VALUES(2,'HOUSTON','LONDON');
INSERT INTO JOURNEY VALUES(3,'LONDON','CHENNAI');
INSERT INTO JOURNEY VALUES(4,'CHENNAI','HOUSTON');
INSERT INTO JOURNEY VALUES(5,'HOUSTON','DALLAS');
INSERT INTO JOURNEY VALUES(6,'SEATTLE','HOUSTON');
INSERT INTO JOURNEY VALUES(7,'NEW YORK','DALLAS');
INSERT INTO JOURNEY VALUES(8,'DALLAS','SAN FRANCISCO');
INSERT INTO JOURNEY VALUES(9,'SAN FRANCISCO','NEW YORK');
INSERT INTO JOURNEY VALUES(10,'SAN FRANCISCO','DALLAS');
INSERT INTO JOURNEY VALUES(11,'DALLAS','SEATTLE');
INSERT INTO JOURNEY VALUES(12,'SEATTLE','SAN FRANCISCO');
INSERT INTO JOURNEY VALUES(13,'LONDON','SAN FRANCISCO');
INSERT INTO JOURNEY VALUES(14,'SEATTLE','DALLAS');
INSERT INTO JOURNEY VALUES(15,'SAN FRANCISCO','SEATTLE');
INSERT INTO JOURNEY VALUES(16,'SAN FRANCISCO','HOUSTON');
```

LEGAL_DOCUMENTS_SET

```
INSERT INTO LEGAL_DOCUMENTS_SET VALUES(1,'PASSPORT OR ANY NATIONAL ID ');
INSERT INTO LEGAL_DOCUMENTS_SET VALUES(2,'PASSPORT AND IMMIGRATION DOCUMENTS');
INSERT INTO LEGAL_DOCUMENTS_SET VALUES(3,'INFANT CARE DOCUMENT');
```

PACKAGE

```
INSERT INTO PACKAGE VALUES(1,1,'EAST WEST TRIP', 3000, 'NEW YORK', 5);
INSERT INTO PACKAGE VALUES(2,2,'EUROPE TRIP', 10000, 'HOUSTON', 10);
INSERT INTO PACKAGE VALUES(3,1,'WEST COAST TRIP', 6000, 'SAN FRANCISCO', 5);
INSERT INTO PACKAGE VALUES(4,2,'WORLD TRIP', 20000, 'HOUSTON', 15);
INSERT INTO PACKAGE VALUES(5,1,'TEXAS TRIP', 1000, 'DALLAS', 5);
```

HOTEL_STAY

```
INSERT INTO HOTEL_STAY VALUES(1,'HUT STAY','HOUSTON');
INSERT INTO HOTEL_STAY VALUES(2,'DAL STAY','DALLAS');
INSERT INTO HOTEL_STAY VALUES(3,'HUT1 STAY','HOUSTON');
INSERT INTO HOTEL_STAY VALUES(4,'LDN STAY','LONDON');
INSERT INTO HOTEL_STAY VALUES(5,'CHN STAY','CHENNAI');
INSERT INTO HOTEL_STAY VALUES(6,'SNF STAY','SAN FRANCISCO');
INSERT INTO HOTEL_STAY VALUES(7,'SEA STAY','SEATTLE');
INSERT INTO HOTEL_STAY VALUES(8,'NY STAY','NEW YORK');
```

COMPRISES_OF

```
INSERT INTO COMPRISES_OF VALUES(2, 1);
INSERT INTO COMPRISES_OF VALUES(6, 1);
INSERT INTO COMPRISES_OF VALUES(4, 2);
INSERT INTO COMPRISES_OF VALUES(6, 2);
INSERT INTO COMPRISES_OF VALUES(6, 3);
INSERT INTO COMPRISES_OF VALUES(7, 3);
INSERT INTO COMPRISES_OF VALUES(4, 4);
INSERT INTO COMPRISES_OF VALUES(5, 4);
INSERT INTO COMPRISES_OF VALUES(3, 5);
```

INCLUDES

```
INSERT INTO INCLUDES VALUES(7, 1);
INSERT INTO INCLUDES VALUES(8, 1);
INSERT INTO INCLUDES VALUES(9, 1);
INSERT INTO INCLUDES VALUES(2, 2);
INSERT INTO INCLUDES VALUES(13, 2);
INSERT INTO INCLUDES VALUES(16, 2);
INSERT INTO INCLUDES VALUES(15, 3);
INSERT INTO INCLUDES VALUES(14, 3);
INSERT INTO INCLUDES VALUES(8, 3);
INSERT INTO INCLUDES VALUES(2, 4);
INSERT INTO INCLUDES VALUES(3, 4);
INSERT INTO INCLUDES VALUES(4, 4);
INSERT INTO INCLUDES VALUES(1, 5);
INSERT INTO INCLUDES VALUES(5, 5);
```

FLIGHT

```
INSERT INTO FLIGHT VALUES(1, 1, 1, INTERVAL '0 09:05' DAY TO MINUTE, INTERVAL '0
14:25' DAY TO MINUTE, 500);
INSERT INTO FLIGHT VALUES(2, 2, 2, INTERVAL '0 08:15' DAY TO MINUTE, INTERVAL '1
09:25' DAY TO MINUTE, 1500);
INSERT INTO FLIGHT VALUES(3, 2, 3, INTERVAL '0 07:05' DAY TO MINUTE, INTERVAL '0
10:25' DAY TO MINUTE, 2500);
INSERT INTO FLIGHT VALUES(4, 2, 4, INTERVAL '0 08:25' DAY TO MINUTE, INTERVAL '1
11:25' DAY TO MINUTE, 1500);
INSERT INTO FLIGHT VALUES(5, 1, 5, INTERVAL '0 09:05' DAY TO MINUTE, INTERVAL '1
12:25' DAY TO MINUTE, 2050);
INSERT INTO FLIGHT VALUES(6, 1, 6, INTERVAL '0 10:05' DAY TO MINUTE, INTERVAL '0
13:25' DAY TO MINUTE, 1350);
INSERT INTO FLIGHT VALUES(7, 1, 7, INTERVAL '0 13:05' DAY TO MINUTE, INTERVAL '0
16:25' DAY TO MINUTE, 1450);
```

```
INSERT INTO FLIGHT VALUES(8, 1, 8, INTERVAL '0 14:05' DAY TO MINUTE, INTERVAL '0
21:25' DAY TO MINUTE, 1500);
INSERT INTO FLIGHT VALUES(9, 1, 9, INTERVAL '0 12:05' DAY TO MINUTE, INTERVAL '1
09:25' DAY TO MINUTE, 700);
INSERT INTO FLIGHT VALUES(10, 1, 10, INTERVAL '0 13:05' DAY TO MINUTE, INTERVAL '0
21:20' DAY TO MINUTE, 1000);
INSERT INTO FLIGHT VALUES(11, 1, 11, INTERVAL '0 06:05' DAY TO MINUTE, INTERVAL '1
20:25' DAY TO MINUTE, 1345);
INSERT INTO FLIGHT VALUES(12, 1, 12, INTERVAL '0 10:05' DAY TO MINUTE, INTERVAL '1
09:00' DAY TO MINUTE, 1500);
INSERT INTO FLIGHT VALUES(13, 2, 13, INTERVAL '0 09:30' DAY TO MINUTE, INTERVAL '1
02:25' DAY TO MINUTE, 1000);
INSERT INTO FLIGHT VALUES(14, 1, 14, INTERVAL '0 09:00' DAY TO MINUTE, INTERVAL '1
00:25' DAY TO MINUTE, 2000);
INSERT INTO FLIGHT VALUES(15, 1, 15, INTERVAL '0 01:05' DAY TO MINUTE, INTERVAL '0
09:25' DAY TO MINUTE, 1234);
INSERT INTO FLIGHT VALUES(16, 1, 16, INTERVAL '0 04:00' DAY TO MINUTE, INTERVAL '0
09:25' DAY TO MINUTE, 1175);
```

BUS

```
INSERT INTO BUS VALUES(1, 1, INTERVAL '0 09:05' DAY TO MINUTE, INTERVAL '0 14:25' DAY
TO MINUTE, 500);
INSERT INTO BUS VALUES(5, 5, INTERVAL '0 09:05' DAY TO MINUTE, INTERVAL '1 12:25' DAY
TO MINUTE, 250);
INSERT INTO BUS VALUES(6, 6, INTERVAL '0 10:05' DAY TO MINUTE, INTERVAL '0 13:25' DAY
TO MINUTE, 350);
INSERT INTO BUS VALUES(7, 7, INTERVAL '0 13:05' DAY TO MINUTE, INTERVAL '0 16:25' DAY
TO MINUTE, 450);
INSERT INTO BUS VALUES(8, 8, INTERVAL '0 14:05' DAY TO MINUTE, INTERVAL '0 21:25' DAY
TO MINUTE, 500);
INSERT INTO BUS VALUES(9, 9, INTERVAL '0 12:05' DAY TO MINUTE, INTERVAL '1 09:25' DAY
TO MINUTE, 600);
INSERT INTO BUS VALUES(10, 10, INTERVAL '0 13:05' DAY TO MINUTE, INTERVAL '0 21:20'
DAY TO MINUTE, 700);
INSERT INTO BUS VALUES(11, 11, INTERVAL '0 06:05' DAY TO MINUTE, INTERVAL '1 20:25'
DAY TO MINUTE, 345);
INSERT INTO BUS VALUES(12, 12, INTERVAL '0 10:05' DAY TO MINUTE, INTERVAL '1 09:00'
DAY TO MINUTE, 500);
INSERT INTO BUS VALUES(14, 14, INTERVAL '0 09:00' DAY TO MINUTE, INTERVAL '1 00:25'
DAY TO MINUTE, 2000);
INSERT INTO BUS VALUES(15, 15, INTERVAL '0 01:05' DAY TO MINUTE, INTERVAL '0 09:25'
DAY TO MINUTE, 234);
INSERT INTO BUS VALUES(16, 16, INTERVAL '0 04:00' DAY TO MINUTE, INTERVAL '0 09:25'
DAY TO MINUTE, 175);
```

TRAIN

```
INSERT INTO TRAIN VALUES(1, 1, INTERVAL '0 09:05' DAY TO MINUTE, INTERVAL '0 14:25'
DAY TO MINUTE, 300);
INSERT INTO TRAIN VALUES(5, 5, INTERVAL '0 09:05' DAY TO MINUTE, INTERVAL '1 12:25'
DAY TO MINUTE, 450);
INSERT INTO TRAIN VALUES(6, 6, INTERVAL '0 10:05' DAY TO MINUTE, INTERVAL '0 13:25'
DAY TO MINUTE, 650);
INSERT INTO TRAIN VALUES(7, 7, INTERVAL '0 13:05' DAY TO MINUTE, INTERVAL '0 16:25'
DAY TO MINUTE, 550);
INSERT INTO TRAIN VALUES(8, 8, INTERVAL '0 14:05' DAY TO MINUTE, INTERVAL '0 21:25'
DAY TO MINUTE, 400);
INSERT INTO TRAIN VALUES(9, 9, INTERVAL '0 12:05' DAY TO MINUTE, INTERVAL '1 09:25'
DAY TO MINUTE, 700);
INSERT INTO TRAIN VALUES(10, 10, INTERVAL '0 13:05' DAY TO MINUTE, INTERVAL '0 21:20'
DAY TO MINUTE, 800);
INSERT INTO TRAIN VALUES(11, 11, INTERVAL '0 06:05' DAY TO MINUTE, INTERVAL '1 20:25'
DAY TO MINUTE, 245);
```

```
INSERT INTO TRAIN VALUES(12, 12, INTERVAL '0 10:05' DAY TO MINUTE, INTERVAL '1 09:00'
DAY TO MINUTE, 300);
INSERT INTO TRAIN VALUES(14, 14, INTERVAL '0 09:00' DAY TO MINUTE, INTERVAL '1 00:25'
DAY TO MINUTE, 450);
INSERT INTO TRAIN VALUES(15, 15, INTERVAL '0 01:05' DAY TO MINUTE, INTERVAL '0 09:25'
DAY TO MINUTE, 294);
INSERT INTO TRAIN VALUES(16, 16, INTERVAL '0 04:00' DAY TO MINUTE, INTERVAL '0 09:25'
DAY TO MINUTE, 185);
```

TRAIN_INSTANCE

```
INSERT INTO TRAIN_INSTANCE VALUES(1, 7, 1, '27-SEP-2014', '29-OCT-2014');
INSERT INTO TRAIN_INSTANCE VALUES(2, 11, 2, '17-OCT-2014', '23-OCT-2014');
INSERT INTO TRAIN_INSTANCE VALUES(3, 5, 5, '20-OCT-2015', '27-OCT-2015');
INSERT INTO TRAIN_INSTANCE VALUES(4, 1, 5, '23-NOV-2015', '29-NOV-2015');
INSERT INTO TRAIN_INSTANCE VALUES(5, 8, 4, '22-AUG-2015', '29-OCT-2015');
INSERT INTO TRAIN_INSTANCE VALUES(6, 9, 4, '29-OCT-2015', '29-OCT-2015');
INSERT INTO TRAIN_INSTANCE VALUES(7, 10, 3, '11-OCT-2015', '13-OCT-2015');
INSERT INTO TRAIN_INSTANCE VALUES(8, 5, 2, '12-OCT-2015', '30-NOV-2015');
INSERT INTO TRAIN_INSTANCE VALUES(9, 9, 1, '13-OCT-2016', '29-OCT-2016');
```

BUS_INSTANCE

```
INSERT INTO BUS_INSTANCE VALUES(1, 1, 2, '17-SEP-2014', '29-OCT-2014');
INSERT INTO BUS_INSTANCE VALUES(2, 9, 5, '7-OCT-2014', '23-NOV-2014');
INSERT INTO BUS_INSTANCE VALUES(3, 5, 1, '2-JUN-2015', '27-JUN-2015');
INSERT INTO BUS_INSTANCE VALUES(4, 8, 5, '2-NOV-2015', '1-DEC-2015');
INSERT INTO BUS_INSTANCE VALUES(5, 7, 3, '22-JUN-2015', '29-JUL-2015');
INSERT INTO BUS_INSTANCE VALUES(6, 5, 4, '20-OCT-2015', '29-NOV-2015');
INSERT INTO BUS_INSTANCE VALUES(7, 10, 2, '12-OCT-2015', '13-NOV-2015');
INSERT INTO BUS_INSTANCE VALUES(8, 9, 4, '12-NOV-2015', '30-NOV-2015');
INSERT INTO BUS_INSTANCE VALUES(9, 11, 4, '14-OCT-2016', '29-NOV-2016');
```

FLIGHT_INSTANCE

```
INSERT INTO FLIGHT_INSTANCE VALUES(1, 9, 4, '17-AUG-2014', '29-SEP-2014');
INSERT INTO FLIGHT_INSTANCE VALUES(2, 11, 2, '7-SEP-2014', '23-OCT-2014');
INSERT INTO FLIGHT_INSTANCE VALUES(3, 1, 5, '2-MAY-2015', '27-JUL-2015');
INSERT INTO FLIGHT_INSTANCE VALUES(4, 7, 1, '2-JAN-2015', '1-NOV-2015');
INSERT INTO FLIGHT_INSTANCE VALUES(5, 8, 4, '22-FEB-2015', '29-MAY-2015');
INSERT INTO FLIGHT_INSTANCE VALUES(6, 9, 1, '20-JAN-2015', '29-OCT-2015');
INSERT INTO FLIGHT_INSTANCE VALUES(7, 10, 3, '12-FEB-2015', '13-FEB-2015');
INSERT INTO FLIGHT_INSTANCE VALUES(8, 5, 4, '12-JAN-2015', '30-SEP-2015');
INSERT INTO FLIGHT_INSTANCE VALUES(9, 8, 3, '14-FEB-2016', '29-JUL-2016');
```

PACKAGE_INSTANCE

```
INSERT INTO PACKAGE_INSTANCE VALUES(1, 3, 2, '17-JUL-2014', '20-SEP-2014');
INSERT INTO PACKAGE_INSTANCE VALUES(2, 4, 2, '7-AUG-2014', '20-OCT-2014');
INSERT INTO PACKAGE_INSTANCE VALUES(3, 1, 4, '2-APR-2015', '20-JUL-2015');
INSERT INTO PACKAGE_INSTANCE VALUES(4, 2, 5, '2-APR-2015', '10-NOV-2015');
INSERT INTO PACKAGE_INSTANCE VALUES(5, 4, 2, '22-APR-2015', '20-MAY-2015');
INSERT INTO PACKAGE_INSTANCE VALUES(6, 3, 4, '20-JUN-2015', '20-OCT-2015');
INSERT INTO PACKAGE_INSTANCE VALUES(7, 2, 3, '12-MAR-2015', '10-SEP-2015');
INSERT INTO PACKAGE_INSTANCE VALUES(8, 1, 1, '12-JUL-2015', '26-SEP-2015');
INSERT INTO PACKAGE_INSTANCE VALUES(9, 5, 4, '14-MAY-2016', '30-JUL-2016');
```

§ 8: Stored Procedures

1) Finding all the transactions a customer has made.

```
CREATE OR REPLACE PROCEDURE CUSTOMER_ALL(CUSTOMER_NO IN CUSTOMER.CUST_ID%TYPE ) AS
--Find for a customer all the travel details he has made.
Cursor TRAIN_TRAVEL
IS
Select  TRAIN_TRANS_ID
from TRAIN_INSTANCE
WHERE CUST_ID = CUSTOMER_NO;
TEMP1 TRAIN_INSTANCE.TRAIN_TRANS_ID%TYPE;
Cursor FLIGHT_TRAVEL
IS
Select  FLIGHT_TRANS_ID
from FLIGHT_INSTANCE
WHERE CUST_ID = CUSTOMER_NO;
TEMP2 FLIGHT_INSTANCE.FLIGHT_TRANS_ID%TYPE;
Cursor BUS_TRAVEL
IS
Select  BUS_TRANS_ID
from BUS_INSTANCE
WHERE CUST_ID =CUSTOMER_NO;
TEMP3 BUS_INSTANCE.BUS_TRANS_ID%TYPE;
Cursor PACKAGE_TRAVEL
IS
Select  PKG_TRANS_ID
from PACKAGE_INSTANCE
WHERE CUST_ID =CUSTOMER_NO;
TEMP4 PACKAGE_INSTANCE.PKG_TRANS_ID%TYPE;

begin
    OPEN TRAIN_TRAVEL;
    FETCH TRAIN_TRAVEL INTO TEMP1;
           DBMS_OUTPUT.put_line('Customer ' || CUSTOMER_NO ||' has booked in train
number ' || TEMP1);
     CLOSE TRAIN_TRAVEL;
     OPEN FLIGHT_TRAVEL;
    FETCH FLIGHT_TRAVEL INTO TEMP2;
          DBMS_OUTPUT.put_line('Customer ' || CUSTOMER_NO ||' has booked in flight
number ' || TEMP2);
     CLOSE FLIGHT_TRAVEL;
      OPEN BUS_TRAVEL;
    FETCH BUS_TRAVEL INTO TEMP3;
     DBMS_OUTPUT.put_line('Customer ' || CUSTOMER_NO ||' has booked in bus number ' ||
TEMP3);
     CLOSE BUS_TRAVEL;
         OPEN PACKAGE_TRAVEL;
    FETCH PACKAGE_TRAVEL INTO TEMP4;
       DBMS_OUTPUT.put_line('Customer ' || CUSTOMER_NO ||' has booked in package
number ' || TEMP4);
     CLOSE PACKAGE_TRAVEL;
END CUSTOMER_ALL;

set serveroutput on;
Begin
CUSTOMER_ALL(3);
end;
```

OUTPUT FOR TRANSACTIONS MADE BY A CUSTOMER:

```
PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.

Customer 3 has booked in train number 7
Customer 3 has booked in flight number 7
Customer 3 has booked in bus number 5
Customer 3 has booked in package number 7
```

2) Find all the international flights provided by the company.

```
CREATE OR REPLACE PROCEDURE INTERNATIONAL_FLIGHT(INTER IN
LEGAL_DOCUMENTS_SET.DOCUMENT_SET_ID%TYPE ) AS
CURSOR FLIGHT_DETAILS
IS
SELECT  FLIGHT_NO
FROM FLIGHT
WHERE DOCUMENT_SET_ID=INTER;
TEMP1 FLIGHT.FLIGHT_NO%TYPE;

BEGIN
    FOR X IN FLIGHT_DETAILS
    LOOP
     TEMP1 := X.FLIGHT_NO;
      DBMS_OUTPUT.PUT_LINE('FLIGHT NUMBER      ' ||   TEMP1 );

    END LOOP;
    EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('NO SUCH FLIGHT EXISTS.');
END INTERNATIONAL_FLIGHT;

SET SERVEROUTPUT ON;
BEGIN
-- INTERNATIONAL FLIGHTS ARE CLASSIFIED BY THE DOCUMENTS REQUIRED FOR THAT FLIGHT.
INTERNATIONAL_FLIGHT(2);
END;
```

OUTPUT FOR INTERNATIONAL FLIGHTS

```
PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.

FLIGHT NUMBER      2
FLIGHT NUMBER      3
FLIGHT NUMBER      4
FLIGHT NUMBER      13
```

3) Find all journeys provided by a package.

```
CREATE OR REPLACE PROCEDURE PACKAGE_JOURNEYS(ID1 IN INCLUDES.PKG_ID%TYPE) AS
CURSOR ALL_JOURNEY
IS
SELECT JOURNEY_ID FROM INCLUDES WHERE PKG_ID = ID1;
TEMP1 INCLUDES.JOURNEY_ID%TYPE;

BEGIN
    FOR X IN  ALL_JOURNEY
    LOOP
     TEMP1 := X.JOURNEY_ID;
      DBMS_OUTPUT.PUT_LINE(' RELATED JOURNEY      ' ||   TEMP1 );
    END LOOP;
    EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('NO SUCH JOURNEY ASSOCIATED WITH THE PACKAGE.');
END PACKAGE_JOURNEYS;

SET SERVEROUTPUT ON;
BEGIN
PACKAGE_JOURNEYS(2);
END;
```

4) Find the costliest bus from a source to destination.

```
CREATE OR REPLACE PROCEDURE COSTLIEST_BUS(SRC JOURNEY.SOURCE%TYPE, DEST
JOURNEY.DESTINATION%TYPE) AS
CURSOR ALL_JOURNEY
IS
SELECT  JOURNEY_ID FROM JOURNEY
WHERE SOURCE = SRC AND DESTINATION = DEST;
TEMP JOURNEY.JOURNEY_ID%TYPE;
MAX_COST1 BUS.BUS_COST%TYPE;
MAX_COST BUS.BUS_COST%TYPE ;

BEGIN
 FOR X IN ALL_JOURNEY
 LOOP
  SELECT MAX(BUS_COST) INTO MAX_COST
        FROM TRAIN
        WHERE JOURNEY_ID = X.JOURNEY_ID;
  DBMS_OUTPUT.PUT_LINE(MAX_COST);
 END LOOP;

END COSTLIEST_BUS;

SET SERVEROUTPUT ON;
BEGIN
COSTLIEST_BUS('HOUSTON','DALLAS');
END;
```

OUTPUT FOR COSTLIEST BUS:

```
PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.

2000
```

5) Find the cheapest flight from a source to destination.

```
CREATE OR REPLACE PROCEDURE CHEAPEST_FLIGHT(SRC JOURNEY.SOURCE%TYPE, DEST
JOURNEY.DESTINATION%TYPE) AS
CURSOR ALL_JOURNEY
IS
SELECT  JOURNEY_ID FROM JOURNEY
WHERE SOURCE = SRC AND DESTINATION = DEST;
TEMP JOURNEY.JOURNEY_ID%TYPE;
MIN_COST1 FLIGHT.FLIGHT_COST%TYPE;
MIN_COST FLIGHT.FLIGHT_COST%TYPE ;

BEGIN
 FOR X IN ALL_JOURNEY
 LOOP
  SELECT MIN(FLIGHT_COST) INTO MIN_COST
        FROM FLIGHT
        WHERE JOURNEY_ID = X.JOURNEY_ID;
  DBMS_OUTPUT.PUT_LINE(MIN_COST);
 END LOOP;

END CHEAPEST_FLIGHT;



SET SERVEROUTPUT ON;
BEGIN
CHEAPEST_FLIGHT('HOUSTON','DALLAS');
```

```
END;
```

OUTPUT FOR CHEAPEST FLIGHT:

```
PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.

700
```

6) FIND THE CHEAPEST TRAIN FROM A GIVEN SOURCE AND DESTINATION

```
CREATE OR REPLACE PROCEDURE CHEAPEST_TRAIN(SRC JOURNEY.SOURCE%TYPE, DEST
JOURNEY.DESTINATION%TYPE) AS
CURSOR ALL_JOURNEY
IS
SELECT  JOURNEY_ID FROM JOURNEY
WHERE SOURCE = SRC AND DESTINATION = DEST;
TEMP JOURNEY.JOURNEY_ID%TYPE;
MIN_COST1 TRAIN.TRAIN_COST%TYPE;
MIN_COST TRAIN.TRAIN_COST%TYPE ;

BEGIN
 FOR X IN ALL_JOURNEY
 LOOP
  SELECT MIN(TRAIN_COST) INTO MIN_COST
        FROM TRAIN
        WHERE JOURNEY_ID = X.JOURNEY_ID;
  DBMS_OUTPUT.PUT_LINE(MIN_COST);
 END LOOP;

END CHEAPEST_TRAIN;


SET SERVEROUTPUT ON;
BEGIN
CHEAPEST_TRAIN('HOUSTON','DALLAS');
END;
```

OUTPUT FOR CHEAPEST TRAIN:

```
PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.

250
```