

# ISAD253.pdf

*anonymous marking enabled*

---

**Submission date:** 01-Jan-2018 10:05AM (UTC+0000)  
**Submission ID:** 80680847  
**File name:** ISAD253\_366585\_1597395560.pdf (2.09M)  
**Word count:** 5362  
**Character count:** 29317

## **CONTENT**

- ❖ **Introduction**
- ❖ **EER Diagram**
- ❖ **Additional Assumptions**
- ❖ **Cardinality Ratio**
- ❖ **Relational Mapping**
- ❖ **Data Normalization**
- ❖ **Data dictionary**
- ❖ **Create Table Statements**
- ❖ **Constraints**
- ❖ **Database Diagram**
- ❖ **Sample records with screen shots**
- ❖ **Create Triggers**
- ❖ **Create Function statements**
- ❖ **Create View statements**
- ❖ **Create Procedure statements**
- ❖ **Critical Appraisal and Comments on future implementation**
- ❖ **Work Load Matrix**
- ❖ **Peer Review Form**

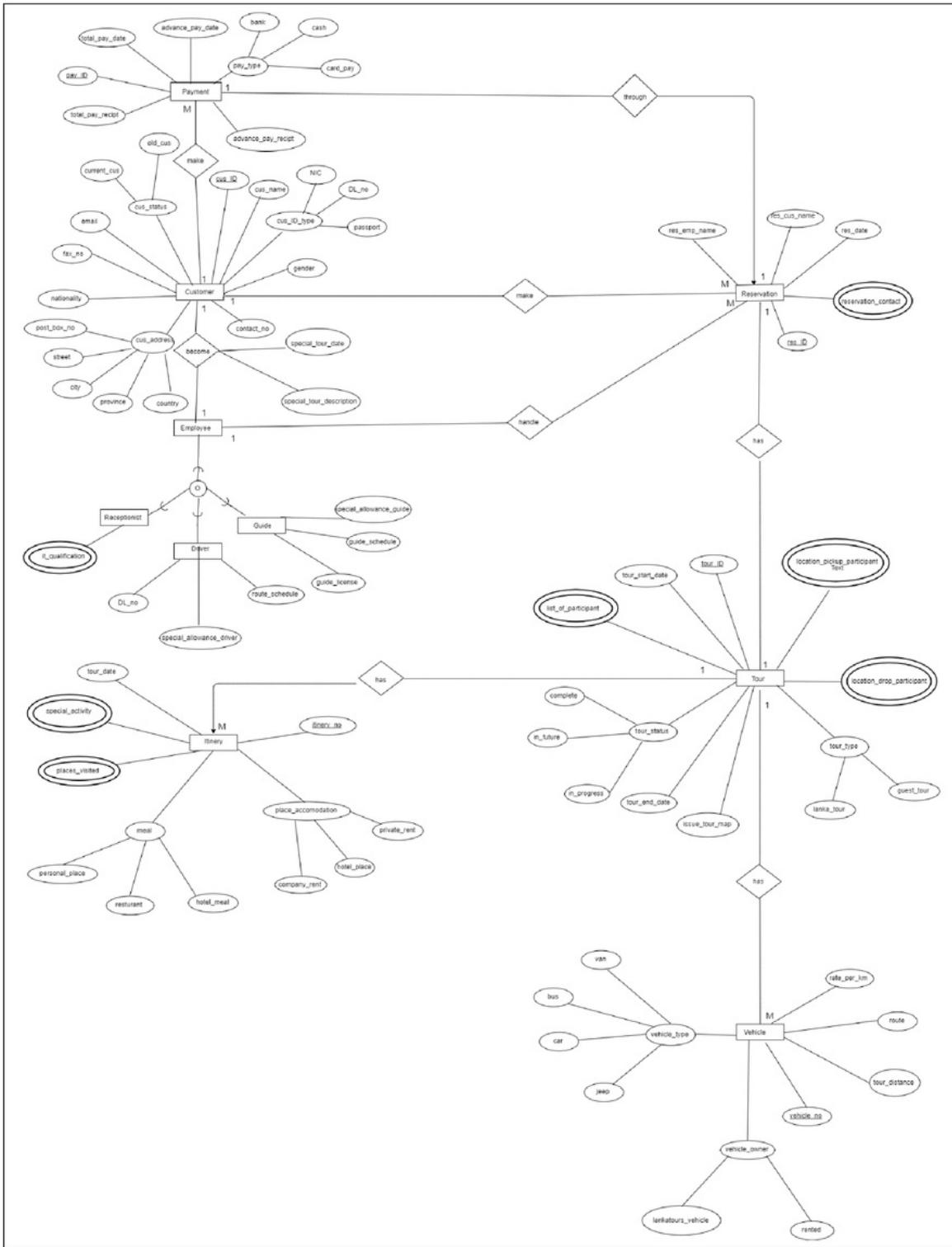
## **Basic introduction about the scenario**

- Lanka Tours is a renowned tour operator in Sri Lanka since 2015. It organic custom private trips to locals and foreigners from individual to large groups.
- Lanka tours needs to implement a new database application to deliver a satisfactory service. They need their system to generate management reports, print customer bills and tour itinerary

We identified seven entities in the above given scenario.

- ✓ Payments are directly linked to the customer, both the reservation payment and the full payment.
- ✓ Reservation process is handled via an employee.
- ✓ Employees are divided to guides, receptionists, drivers. Sometimes one employee can serve two purposes.
  - i.e. guide and driver both.
- ✓ There is a possibility of an employee becoming a customer as well.

# **EER-Diagram**



## **Additional Assumptions**

- ❖ Vehicle rate per kilometer depends on vehicle type.
- ❖ The same person makes the reservation and payment.
- ❖ The reservation should be produced, when making the advance payment.
- ❖ Employee should be a local resident.
- ❖ The pickup and drop location of a foreign can be the airport.
- ❖ Employee can also be a customer. We assume as an entity, “Special” in EER-Diagram.
- ❖ Customer contact number, fax-no and email can be possible empty.
- ❖ Assume that tour-type can be made by Lanka tours or the guest of their interests.
- ❖ Each guide is paid Rs5,000 per day and an additional 5% allowance of the total tour package.
- ❖ An additional 3% allowance of the total tour-package is paid for the drivers and the vehicle.
- ❖ When making reservation, guests must pay an advance payment of the 50% of their tour package.

## **Cardinality Ratios**

1. One payment is relevant for one reservation (1:1)
2. One customer can have many payments(1:M)
3. One employee can become a customer (1:1)
4. One employee can handle many reservations(1:M)
5. Each reservation can have only one tour (1:1)
6. One tour can have many vehicles(1:M)
7. Each tour has more than one itinerary(1:M)
8. One customer can make many reservations(1:M)

## Relational Mapping

### Customer

<u>cus_ID</u>	cus_name	gender	nationality	NIC	DL_no	passport
---------------	----------	--------	-------------	-----	-------	----------

old_cus	current_cus	email	P.O box	street
---------	-------------	-------	---------	--------

city	state	country	contact_no	fax_no
------	-------	---------	------------	--------

### Payment

FK

<u>pay_ID</u>	<u>cus_ID</u>	bank	cash	card	advance_pay_date
---------------	---------------	------	------	------	------------------

total_pay_date	total_pay_receipt	advance_pay_receipt	pay_cus_name
----------------	-------------------	---------------------	--------------

FK

old_cus	current_cus	email	P.O box	street
---------	-------------	-------	---------	--------

city	state	country	contact_no	fax_no
------	-------	---------	------------	--------

### Payment

FK

<u>pay_ID</u>	<u>cus_ID</u>	bank	cash	card	advance_pay_date
---------------	---------------	------	------	------	------------------

total_pay_date	total_pay_receipt	advance_pay_receipt	pay_cus_name
----------------	-------------------	---------------------	--------------

FK

<u>tour_end_date</u>	issue_tour_map	complete	inprogress	in_future
----------------------	----------------	----------	------------	-----------

### Tour Participant

<u>tour_ID</u>	list_of_participant
----------------	---------------------

### Tour Location

<u>tour_ID</u>	location_pickup_participant	location_drop_participant
----------------	-----------------------------	---------------------------

### Vehicle

FK

<u>vehicle_ID</u>	<u>tour_ID</u>	route	tour_distance	rate_per_km
-------------------	----------------	-------	---------------	-------------

van	car	bus	jeep	rented	vehicle_lanka_tour
-----	-----	-----	------	--------	--------------------

### Itinerary

FK

<u>Itinerary_no</u>	<u>tour_ID</u>	tour_date	private_rent	company_rent
---------------------	----------------	-----------	--------------	--------------

<u>hotel_place</u>	<u>Personal_place</u>	<u>resturant</u>	<u>hotel_for_meal</u>
--------------------	-----------------------	------------------	-----------------------

**Itinerary Activities**

<u>Itinerary no</u>	<u>Special activity</u>

**Itinerary Visited**

<u>Itinerary no</u>	<u>places_visited</u>

**Employee**

<u>emp ID</u>	<u>emp_name</u>	<u>gender</u>	<u>address</u>	<u>salary</u>	<u>contact_no</u>

**Emp Recep**

<u>emp ID</u>	<u>IT_qualification</u>

**Emp Driver**

<u>emp ID</u>	<u>DL_no</u>	<u>driver_special_allowance</u>	<u>route_scedule</u>

**Emp Guide**

<u>emp ID</u>	<u>guide_license_no</u>	<u>guide_schedule</u>	<u>guide_special_allowance</u>

**Employee Customer**

<u>emp ID</u>	<u>cus_ID</u>	<u>special_tour_no</u>	<u>special_tour_date</u>

## Data Normalization

### Customer

<u>cus_ID</u>	<u>cus_name</u>	<u>gender</u>	<u>nationality</u>	<u>email</u>	<u>fax_no</u>

### 1NF

- ✓ There are no repeating groups.
- ✓ No composite attributes.
- ✓ No multi-valued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

Cus\_ID → {cus\_name, gender, nationality, email, fax\_no}

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies Therefore, this table is in 3NF.

### Cus\_ID Type

<u>cus_ID</u>	NIC	DL_no	passport
	↑	↑	↑

### 1NF

- ✓ There are no repeating groups.
- ✓ There is a composite attribute therefore it should be in 1NF.
- ✓ No multi-valued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{Cus\_ID} \rightarrow \{\text{NIC, DL_no , passport}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Cus\_Status

<u>cus_ID</u>	<u>old_cus</u>	<u>current_cus</u>

### 1NF

- ✓ There are no repeating groups.
- ✓ No composite attributes.
- ✓ There is multi-valued attribute therefore it should convert to 1NF.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

**Cus\_ID**→ { **old\_cus**, **current\_cus** }

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Cus Address

<b>cus_ID</b>	<b>post_box_no</b>	<b>street</b>	<b>city</b>	<b>province</b>	<b>country</b>

## 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ There is composite attribute therefore it should be in 1NF.
- ✓ The attributes are atomic and single valued.

## Functional Dependencies

$\text{Cus\_ID} \rightarrow \{\text{post\_box\_no}, \text{street}, \text{city}, \text{province}, \text{country}\}$

## 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Payment

<u>Pay_ID</u>	<u>cus_ID</u>	<u>advance_pay_date</u>	<u>total_pay_date</u>	<u>advance_pay_receipt</u>	<u>res_ID</u>	<u>pay_cus_name</u>

## 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ No composite attributes.
- ✓ This table is in 1NF.
- ✓ The attributes are single and atomic.

## Functional Dependencies

$\text{Cus\_ID} \rightarrow \{ \text{cus\_ID}, \text{advance\_pay\_date}, \text{total\_pay\_date}, \text{advance\_pay\_receipt}, \text{res\_ID}, \text{pay\_cus\_name} \}$

## 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Pay Customer

<u>pay_ID</u>	<u>cus_ID</u>	<u>pay_cus_name</u>

### 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ No composite attributes.
- ✓ This table is in 1NF.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{pay\_ID} \rightarrow \{\text{cus\_ID}, \text{pay\_cus\_name}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Pay Type

<u>pay ID</u>	bank	cash	card
	↑	↑	↑

## 1NF

- ✓ There are no repeating groups.
- ✓ There is composite attribute therefore it should be in 1NF.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.

## Functional Dependencies

**Cus\_ID → { bank, cash, card }**

## 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Reservation Contact

<u>res_ID</u>	<u>res_cus_contact_no</u>

### 1NF

- ✓ There are no repeating groups.
- ✓ This is a multivalued attribute therefore it should be in 1NF.
- ✓ No nested relations.
- ✓ No composite attributes.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{res\_ID} \rightarrow \{\text{res\_cus\_contact\_no}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Reservation

res_ID	emp_ID	cus_ID	res_date	res_cus_name	res_emp_name

### 1NF

- ✓ There are no repeating groups.
- ✓ No nested relations.
- ✓ No multivalued attributes.
- ✓ No composite attributes.
- ✓ This table is in 1NF.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{res\_ID} \rightarrow \{\text{emp\_ID}, \text{cus\_ID}, \text{res\_date}, \text{res\_cus\_name}, \text{res\_emp\_name}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Reservation Customer

<u>res_ID</u>	<u>Cus_ID</u>	<u>Res_cus_name</u>

### 1NF

- ✓ There are no repeating groups.
- ✓ No nested relations.
- ✓ No multivalued attributes.
- ✓ No composite attributes.
- ✓ This table is in 1NF.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{res\_ID} \rightarrow \{\text{cus\_ID}, \text{res\_cus\_name}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Reservation Employee

res_ID	emp_ID	Res_emp_name

### 1NF

- ✓ There are no repeating groups.
- ✓ No nested relations.
- ✓ No multivalued attributes.
- ✓ No composite attributes.
- ✓ This table is in 1NF.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{res\_ID} \rightarrow \{\text{emp\_ID}, \text{res\_emp\_name}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Employee

<u>emp_ID</u>	<u>emp_name</u>	<u>gender</u>	<u>address</u>	<u>salary</u>

## 1NF

- ✓ There are no repeating groups.
- ✓ No nested relations.
- ✓ No multivalued attributes.
- ✓ No composite attributes.
- ✓ This table is in 1NF.
- ✓ The attributes are atomic and single valued.

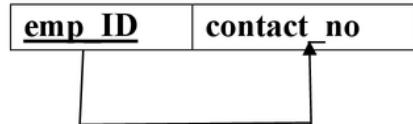
## Functional Dependencies

$\text{emp\_ID} \rightarrow \{\text{emp\_name}, \text{gender}, \text{address}, \text{salary}\}$

## 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Employee Contact



### 1NF

- ✓ There are no repeating groups.
- ✓ There is a multivalued attribute therefore, it should be in 1NF.
- ✓ No composite attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{emp\_ID} \rightarrow \{\text{contact\_no}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Emp\_ Receptionist

<u>emp_ID</u>	It_qualification

### 1NF

- ✓ There are no repeating groups.
- ✓ There is a multivalued attribute therefore, it should be in 1NF.
- ✓ No composite attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{emp\_ID} \rightarrow \{ \text{It\_qualification} \}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

### Emp\_Driver

<b>emp_ID</b>	<b>DL_no</b>	<b>Special_allowance</b>	<b>route_schedule</b>

### 1NF

- ✓ There are no repeating groups.
- ✓ No composite attributes.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.

### Functional Dependencies

**emp\_ID** → { **DL\_no** , **Special\_allowance** , **rout\_schedule** }

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Emp\_Guide

<u>emp_ID</u>	<u>guide_license</u>	<u>guide_schedule</u>	<u>special_allowance</u>

### 1NF

- ✓ There are no repeating group.
- ✓ This table is in 1NF.
- ✓ No multivalued attributes.
- ✓ No composite attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{emp\_ID} \rightarrow \{\text{guide\_license}, \text{guide\_schedule}, \text{guide\_allowance}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Employee Customer

<u>emp_ID</u>	<u>cus_ID</u>	<u>special_tour_description</u>	<u>special_tour_date</u>

### 1NF

- ✓ There are no repeating group.
- ✓ This table is in 1NF.
- ✓ No multivalued attributes.
- ✓ No composite attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

$\text{emp\_ID} \rightarrow \{\text{cus\_ID}, \text{special\_tour\_descrition}, \text{special\_tour\_date}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Itinerary

<u>itinerary_ID</u>	<u>tour_dates</u>

## 1NF

- ✓ There are no repeating group.
- ✓ No multivalued attributes.
- ✓ No composite attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.

## Functional Dependencies

**itinerary\_no** → {tour\_date }

## 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Itinerary Meal

<u>itinerary_no</u>	<u>hotel_meal</u>	<u>restaurant</u>	<u>personal_place</u>
↑ ↑ ↑			

### 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.
- ✓ This is a composite attribute therefore it should convert to 1NF.

### Functional Dependencies

**itinerary\_no → { hotel\_meal, restaurant , personal\_place }**

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Itinerary Place Accommodation

<u>itinerary_no</u>	<u>private_rent</u>	<u>hotel_place</u>	<u>company_rent</u>

### 1NF

- ✓ There are no repeating group.
- ✓ There is a composite attribute therefore, it should be in 1NF.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

itinerary\_no → { private\_rent , hotel\_place , company\_rent }

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Itinerary Special Activity

<u>itinerary_no</u>	<u>special_activities</u>

### 1NF

- ✓ There are no repeating group.
- ✓ There is a multivalued attribute therefore, it should be in 1NF.
- ✓ No nested relations.
- ✓ No composite attributes.
- ✓ The attributes are atomic and single valued.

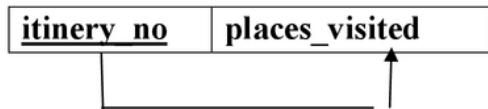
### Functional Dependencies

$\text{itinerary\_no} \rightarrow \{\text{special\_activity}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Itinerary\_Places Visited



### 1NF

- ✓ There are no repeating group.
- ✓ There is a multivalued attribute therefore, it should be in 1NF.
- ✓ No nested relations.
- ✓ No composite attributes.
- ✓ The attributes are atomic and single valued.

### Functional Dependencies

**itinerry\_no** → { places\_visited }

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Vehicle

<u>vehicle_no</u>	<u>tour_ID</u>	<u>tour_distance</u>	<u>rate_per_km</u>	<u>route</u>

### 1NF

- ✓ There are no repeating group.
- ✓ There is a multivalued attribute therefore, it should be in 1NF.
- ✓ No nested relations.
- ✓ No composite attributes.
- ✓ The attributes are atomic and single valued.

### Functional Dependency

$\text{vehicle\_no} \rightarrow \{\text{tour\_ID}, \text{tour\_distance}, \text{rate\_per\_km}, \text{route}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

### Vehicle Type

<u>vehicle_no</u>	<u>tour_ID</u>	<u>tour_distance</u>	<u>rate_per_km</u>	<u>route</u>

### 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.
- ✓ This is a composite attribute therefore it should convert to 1NF.

### Functional Dependency

**vehicle\_no** → { tour\_ID, tour\_distance, rate\_per\_km , route}

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

### Vehicle Own

<b><u>vehicle_no</u></b>	<b><u>rented_vehicle</u></b>	<b><u>Lanka_tours_vehicle</u></b>
--------------------------	------------------------------	-----------------------------------

### **1NF**

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.
- ✓ This is a composite attribute therefore it should convert to 1NF.

### **Functional Dependency**

**vehicle\_no** → { **rented\_vehicle** , **lanka\_tours** }

### **2NF|3NF**

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Tour

<u>tour_ID</u>	<u>reservation_ID</u>	<u>tour_start_date</u>	<u>tour_end_date</u>	<u>issue_tour_map</u>

### 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.
- ✓ This is a composite attribute therefore it should convert to 1NF.

### Functional Dependency

$\text{tour\_ID} \rightarrow \{\text{reservation\_ID}, \text{tour\_start\_date}, \text{tour\_end\_date}, \text{issue\_tour\_map}\}$

### 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Tour Type

<u>tour_ID</u>	<u>lanka_tour</u>	<u>guest_tour</u>

## 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.
- ✓ This is a composite attribute therefore it should convert to 1NF.

## Functional Dependency

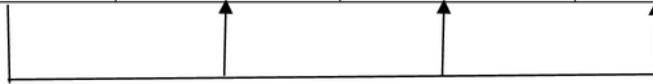
$\text{tour\_ID} \rightarrow \{\text{lanka\_tour}, \text{guest\_tour}\}$

## 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Tour Status

<u>tour_ID</u>	complete	inprogress	in_future



## 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.
- ✓ This is a composite attribute therefore it should convert to 1NF.

## Functional Dependency

$\text{tour\_ID} \rightarrow \{\text{complete}, \text{inprogress}, \text{in\_future}\}$

## 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Tour Participants

<u>tour_ID</u>	list_of_participants

## 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.
- ✓ This is a composite attribute therefore it should convert to 1NF.

## Functional Dependency

$\text{tour\_ID} \rightarrow \{\text{list\_of\_participants}\}$

## 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## Tour Location

<b>tour_ID</b>	<b>location_pickup_participants</b>	<b>location_drop_participants</b>

## 1NF

- ✓ There are no repeating groups.
- ✓ No multivalued attributes.
- ✓ No nested relations.
- ✓ The attributes are atomic and single valued.
- ✓ This table is in 1NF.
- ✓ This is a composite attribute therefore it should convert to 1NF.

## Functional Dependency

$\text{tour\_ID} \rightarrow \{\text{location\_pickup\_participants}, \text{location\_drop\_participants}\}$

## 2NF|3NF

- ✓ There are no partial dependencies and transitive dependencies therefore, this table is in 3NF.

## **Data Dictionary**

### **Customer**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
cus_Id		bigint	Customer Id of the customer
cus_name	255	Varchar	Customer name of the customer
gender	1	Varchar	Gender of the customer
nationality	45	Varchar	Nationality of customer
Email	250	Varchar	Email of the customer
fax_no	10	int	Fax number of the customer
contact_no	15	int	Contact number of the customer

### **Customer\_Id\_type**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
cus_ID		bigint	Customer id of customer Id type
NIC	10	Varchar	NIC of the customer Id type
DI_no	8	varchar	DI_no number of the customer Id type
Passport	8	Varchar	Passport number of the Id type

### **Cus\_Status**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
cus_ID		bigint	Customer Id of customer status
old_cus	45	varchar	Old customer of customer status
current_cus	45	Varchar	Current customer of customer status

### **Customer\_Address**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
cus_ID		bigint	Customer Id of customer address
post_box_no	10	Int	Post box of customer address
street	45	Varchar	Street of customer address
city	50	Varchar	City of customer address
state	50	Varchar	State of customer address
country	50	Varchar	Country of customer address

## **Payment**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
pay_ID		bigint	Payment Id of payment
advance_pay_date		Date	Advance pay date of payment
total_pay_date		Date	Total payment of payment
total_pay_receipt	45	Varchar	Total pay receipt of payment
advance_pay_receipt	45	Varchar	Advance payment receipt of payment
pay_cus_name	255	Varchar	Payment customer name of payment
reservation_Id		int	Reservation id of payment
Cus_Id		Int	Customer Id of payment

## **Pay\_Type**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
pay_ID		bigint	Payment Id of payment type
bank	45	Varchar	Bank of payment type
Cash	45	Varchar	Cash of payment type
Card_pay	45	Varchar	Card of payment type

### **Pay\_Customer**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>Description</b>
Pay_ID		bigint	Payment ID of payment table
Cus_ID		bigint	Customer ID of customer table
Pay_cus_name	255	varchar	Payment customer name

### **Reservation\_contact**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
res_ID		bigint	Reservation Id of reservation table
res_cus_contact_no	15	Int	Reservation customer contact number of reservation contact

### **Reservation**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
res_ID		int	Reservation Id of reservation
res_cus_name	255	Varchar	Reservation customer name of reservation
res_emp_name	255	Varchar	Reservation employee name of reservation
emp_Id		Int	Employee id of reservation
cus_Id		Int	Customer id of reservation
res_date		Date	Reservation date of reservation

### **Reservation\_Customer**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
res_ID		bigint	Reservation ID of reservation table
Cus_ID		bigint	Customer ID of customer table
res_cus_name	255	varchar	Reservation customer name

### **Reservation\_Employee**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
Res_ID		bigint	Reservation ID of reservation table
Emp_ID			Employee ID of employee table
Res_emp_name	255	varchar	Reservation employee name

### **Employee**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
emp_id		bigint	Employee id of employee
emp_name	255	Varchar	Employee name of employee
gender	1	Varchar	Gender of employee
address	255	Varchar	Address of the employee
salary		Int	Salary of the employee

### **Employee\_Contact**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
emp_Id		bigint	Employee Id of employee contact
emp_contact_no	10	Int	Employee contact number of employee contact

### **Emp\_Receptionist**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
emp_Id		Int	Employee id of employee receptionist
It_qualification	255	Varchar	It qualification of employee receptionist

### **Emp\_Driver**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
emp_Id		bigint	Employee Id of employee driver
DL_no	8	Varchar	Driving license number of employee driver
special_allowance_driver	25	Int	Special allowance of employee driver
route_schedule	255	Varchar	Route schedule of employee driver

### **Emp\_Guide**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
emp_Id		bigint	Employee Id of employee guide
guide_license	45	Varchar	Guide license of employee driver
guide_schedule	255	Varchar	Guide schedule of employee driver
special_allowance_guide		Int	Special allowance of employee guide

### **Employee\_Customer**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
emp_Id		bigint	Employee ID of employee table
Cus_ID		bigint	Customer ID of customer table
Special_tour_description	255	varchar	Special tour description about customer as employee
Special_tour_date		date	Special tour dates about customer as employee

### **Itinerary**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
itinerary_no		bigint	Itinerary ID of itinerary
tour_date		Date	Tour dates of itinerary

### **Itinerary\_meal**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
itinerary_id		Int	Itinerary Id of itinerary meal
hotel_meal	255	Varchar	Hotel meal of itinerary meal
restaurant	255	Varchar	Restaurant of itinerary meal
personal_place	255	Varchar	Personal place of itinerary meal

### **Itinerary\_place\_accomadation**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
itinerary_id		bigint	Itinerary Id of itinerary table
private_rent	255	Varchar	Private rent of itinerary place accommodation
hotel_place	255	Varchar	Hotel place of itinerary place accommodation
company_rent	255	Varchar	Company rent of itinerary place accommodation

### **Itinerary\_Places\_Visited**

<b>Field name</b>	<b>Filed size</b>	<b>Data type</b>	<b>description</b>
itinerary_id		bigint	Itinerary Id of itinerary table
places_visited	255	varchar	Visited places of itinerary

### **Itinerary\_Special\_Activity**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>description</b>
itinerary_Id		bigint	Itinerary Id of itinerary table
Special_activity	255	varchar	Special activities of itinerary

### **TOUR**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>description</b>
Tour_ID		bigint	Tour ID of the tour table
tour_start_date		date	Tour start dates of tour table
tour_end_date		date	Tour end dates of tour table
tour_end_date		date	Tour end dates of tour table

### **Tour\_Location**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>description</b>
Tour_ID		bigint	Tour ID of the tour table
location_pickup_participant	255	varchar	Tour location pickup participant
location_drop_participant	255	varchar	Tour location drop participant

### **Tour\_Participant**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>description</b>
Tour_ID		bigint	Tour ID of the tour table
list_of_participant	255	varchar	Tour list of participants

### **Tour\_Status**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>description</b>
Tour_ID		bigint	Tour ID of the tour table
in_future	255	varchar	In future tours description
In_progress	255	varchar	Current tours description
completed	255	varchar	Completed tours description

### **Tour\_Type**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>description</b>
Tour_ID		bigint	Tour ID of the tour table
lanka_tours_des	255	varchar	Tour made by lanka_tours description
guest_tour_des	255	varchar	Tour made customer choices description

### **Vehicle**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>description</b>
Vehicle_no		bigint	Vehicle number of vehicle table
tour_distance	255	varchar	distance of tour
route_description	255	varchar	Route description of tour
rate_per_km	(5,2)	decimal	Rate_per_km for tour

### **Vehicle\_Own**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>description</b>
Vehicle_no		bigint	Vehicle number of vehicle table
rented_vehicle	45	varchar	Tour vehicle rented description
lanka_tours_vehicle	45	varchar	Lanka tours vehicle description

### **Vehicle\_Type**

<b>Field name</b>	<b>Field size</b>	<b>Data type</b>	<b>description</b>
Vehicle_no		bigint	Vehicle number of vehicle table
car	45	varchar	Vehicle type using for tour
van	45	varchar	Vehicle type using for tour
bus	45	varchar	Vehicle type using for tour
jeep	45	varchar	Vehicle type using for tour

9

## Create Table Statements

### **Customer Table**

```
create table CUSTOMER(  
    cus_ID bigint not null,  
    cus_name varchar(255),  
    gender varchar(1),  
    nationality varchar(45),  
    email varchar(250),  
    fax_no int,  
    contact_no int);
```

### **Cus\_Address Table**

```
create table CUS_ADDRESS(  
    cus_ID bigint not null,  
    post_box_no int,  
    street varchar(45),  
    city varchar(50),  
    state varchar(50),  
    country varchar(50));
```

### **Cus\_Id\_Type Table**

```
create table CUS_ID_TYPE(  
    cus_ID bigint not null,  
    NIC varchar(10),  
    DL_no varchar(8),  
    passport varchar(8));
```

### **Cus\_Status Table**

```
create table CUS_STATUS(  
    cus_ID bigint not null,  
    old_cus varchar(45)  
    current_cus varchar(45));
```

### **Employee Table**

2

```
create table EMPLOYEE(  
    emp_ID bigint not null,  
    emp_name varchar(255),  
    gender varchar(1),  
    address varchar(255),  
    salary int);
```

6

### **Emp\_Driver Table**

```
create table EMP_DRIVER(  
    emp_ID bigint not null,  
    DL_no varchar(8),  
    special_allowance_driver int,  
    route_schedule varchar(255));
```

6

### **Emp\_Guide Table**

```
create table EMP_GUIDE(  
    emp_ID bigint not null,  
    guide_license varchar(45),  
    guide_schedule varchar(255),  
    special_allowance_guide int);
```

## **6** Emp\_Receptionist Table

```
create table EMP_RECEPTIONIST(  
    emp_ID bigint not null,  
    it_qualification varchar(255));
```

## **12** Employee\_Customer Table

```
Create table EMPLOYEE_CUSTOMER(  
    emp_ID bigint not null,  
    cus_ID bigint not null,  
    special_tour_description varchar(255);  
    special_tour_date date);
```

## Employee\_Contact Table

```
7 create table EMPLOYEE_CONTACT(  
    emp_ID bigint not null,  
    emp_contact_no int);
```

## Itinerary Table

```
create table ITINERY(  
    itinerary_ID bigint not null,  
    tour_date date);
```

### **Itinerary\_Meal Table**

```
create table ITINERY_MEAL(  
    [6]  
    itinerary_ID bigint not null,  
    hotel_meal varchar(255),  
    restaurant varchar(255),  
    personal_place varchar(255));
```

### **Itinerary\_Place\_Accomodation Table**

```
create table ITINERY_PLACE_ACCOMODATION(  
    [6]  
    itinerary_ID bigint not null,  
    private_rent varchar(255),  
    hotel_place varchar(255),  
    company_rent varchar(255));
```

### **Itinerary\_Place\_Visited Table**

```
create table ITINERY_PLACES_VISITED(  
    itinerary_ID bigint not null,  
    places_visited varchar(255));
```

### **Itinerary\_Special\_Activity Table**

```
create table ITINERY_SPECIAL_ACTIVITY(  
    itinerary_ID bigint not null,  
    special_activity varchar(255));
```

### **Payment Table**

```
create table PAYMENT(  
    pay_ID bigint not null,  
    total_pay_receipt varchar(45),  
    advance_pay_receipt varchar(45),  
    total_pay_date date,  
    advance_pay_date date);
```

### **Pay\_Customer Table**

```
Create table PAY_CUSTOMER(  
    pay_ID bigint not null,  
    cus_ID bigint not null,
```

### **Pay\_Type Table**

```
Create table PAY_TYPE(  
    pay_ID bigint not null,  
    bank varchar(45),  
    cash varchar(45),  
    card_pay varchar(45));
```

### **Reservation Table**

```
create table RESERVATION(  
    res_ID bigint not null,  
    emp_ID bigint not null,  
    cus_ID bigint not null,  
    res_date date);
```

### **Reservation\_Customer Table**

```
2
create table RESERVATION_CUSTOMER(
    res_ID bigint not null,
    cus_ID bigint not null,
    res_cus_name varchar(255),
    constraint PK_Reservation_Cus primary key (res_ID,cus_ID));
```

### **Reservation\_Employee Table**

```
7
create table RESERVATION_EMPLOYEE(
    res_ID bigint not null,
    emp_ID bigint not null,
    res_emp_name varchar(255),
    constraint PK_Reservation_Emp primary key (res_ID,emp_ID));
```

### **Reservation\_Contact Table**

```
create table RESERVATION_CONTACT(
    res_ID bigint not null,
    res_cus_contact_no int);
```

### **Tour Table**

```
create table TOUR(
    tour_ID bigint not null,
    tour_start_date date,
    tour_end_date date);
```

### **Tour\_Location Table**

```
create table TOUR_LOCATION(  
tour_ID bigint not null,  
location_pickup_participant varchar(255),  
location_drop_participant varchar(255));
```

### **Tour\_Participant Table**

```
create table TOUR_PARTICIPANT(  
tour_ID bigint not null,  
list_of_participant varchar(255));
```

### **Tour\_Status Table**

```
create table TOUR_STATUS(  
tour_ID bigint not null,  
in_future varchar(255),  
in_progress varchar(255),  
completed varchar(255));
```

6

### **Tour\_Type Table**

```
create table TOUR_TYPE(  
tour_ID bigint not null,  
lanka_tours_des varchar(255),  
guest_tour_des varchar(255));
```

**Vehicle Table**

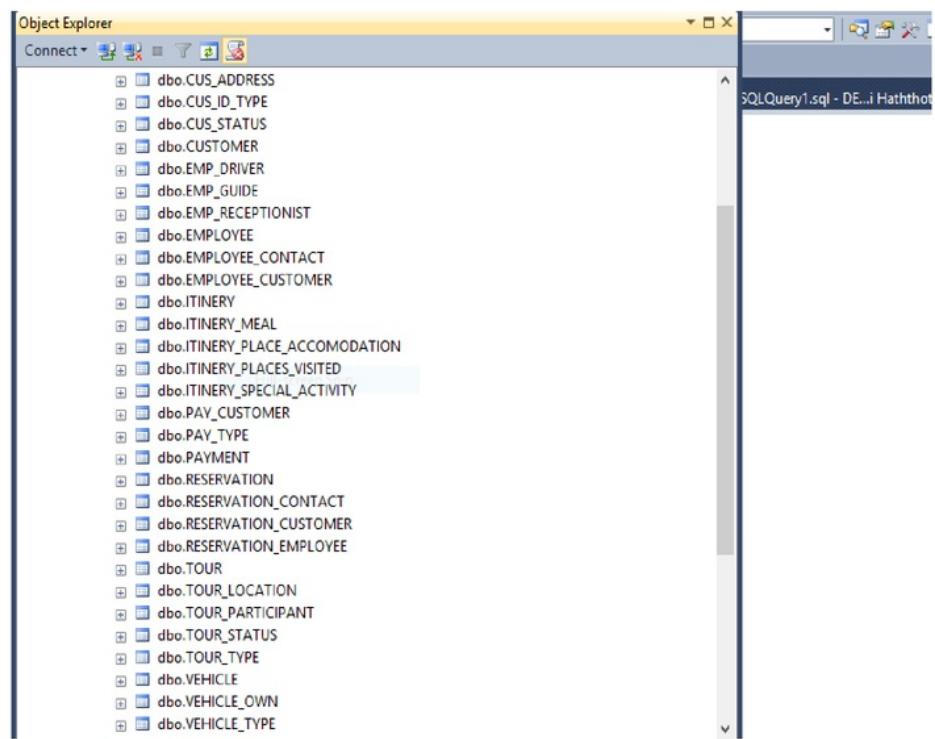
```
create table VEHICLE(  
    vehicle_NO bigint not null,  
    tour_distance varchar(255),  
    route_description varchar(255),  
    rate_per_km decimal(5,2));
```

**Vehicle\_Own Table**

```
create table VEHICLE_OWN(  
    vehicle_NO bigint not null,  
    rented_vehicle varchar(45),  
    lanka_tours_vehicle varchar(45));
```

**Vehicle\_Type Table**

```
create table VEHICLE_TYPE(  
    vehicle_NO bigint not null,  
    car varchar(45),  
    van varchar(45),  
    bus varchar(45),  
    jeep varchar(45));
```



## **CONSTRAINTS**

### **Foreign key constraints**

4

ALTER TABLE EMPLOYEE

ADD FOREIGN KEY (cus\_ID) REFERENCES CUSTOMER(cus\_ID);

ALTER TABLE ITINERY

ADD FOREIGN KEY (tour\_ID) REFERENCES TOUR(tour\_ID);

ALTER TABLE PAYMENT

ADD FOREIGN KEY (cus\_ID) REFERENCES CUSTOMER(cus\_ID);

ALTER TABLE RESERVATION

ADD FOREIGN KEY (emp\_ID) REFERENCES EMPLOYEE(emp\_ID);

ALTER TABLE TOUR

ADD FOREIGN KEY (res\_ID) REFERENCES RESERVATION(res\_ID);

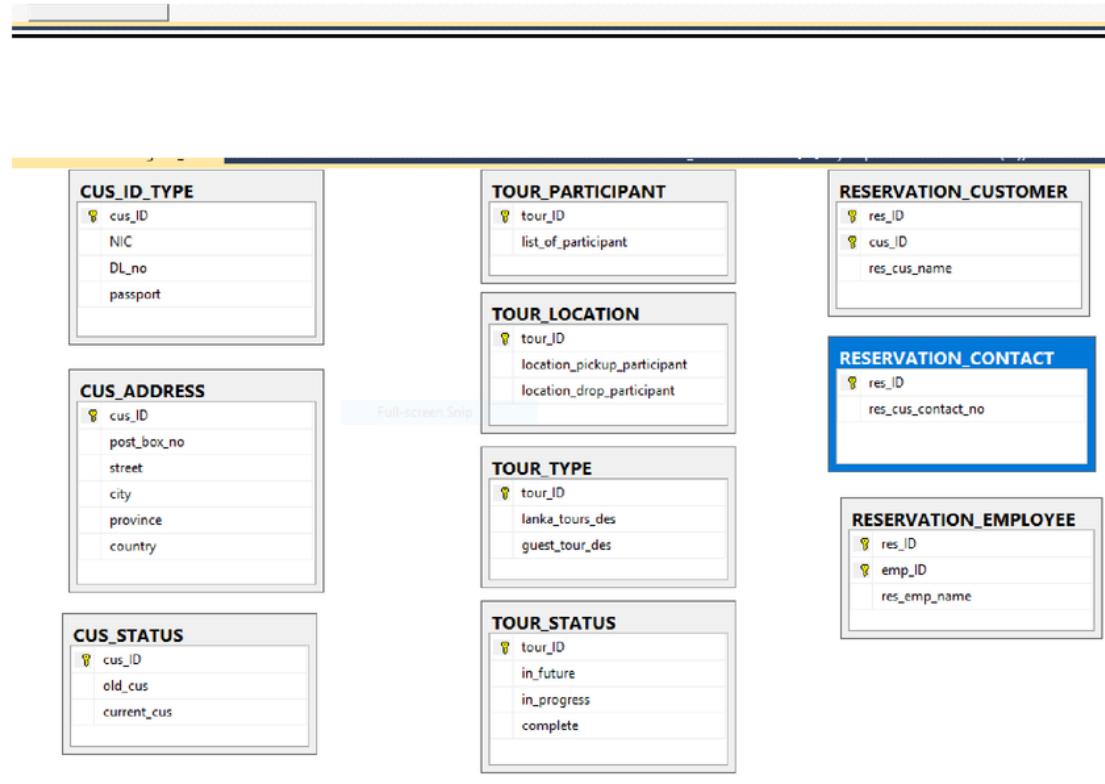
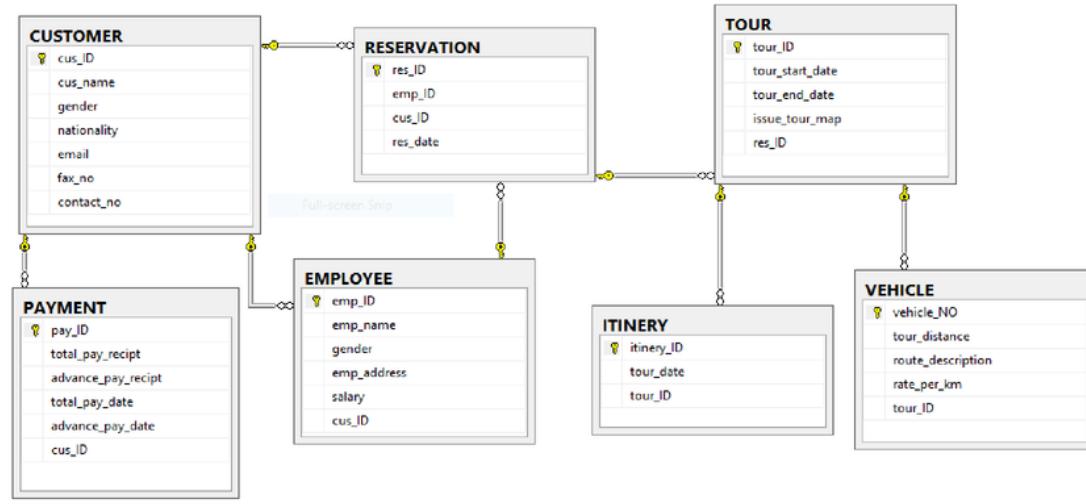
ALTER TABLE VEHICLE

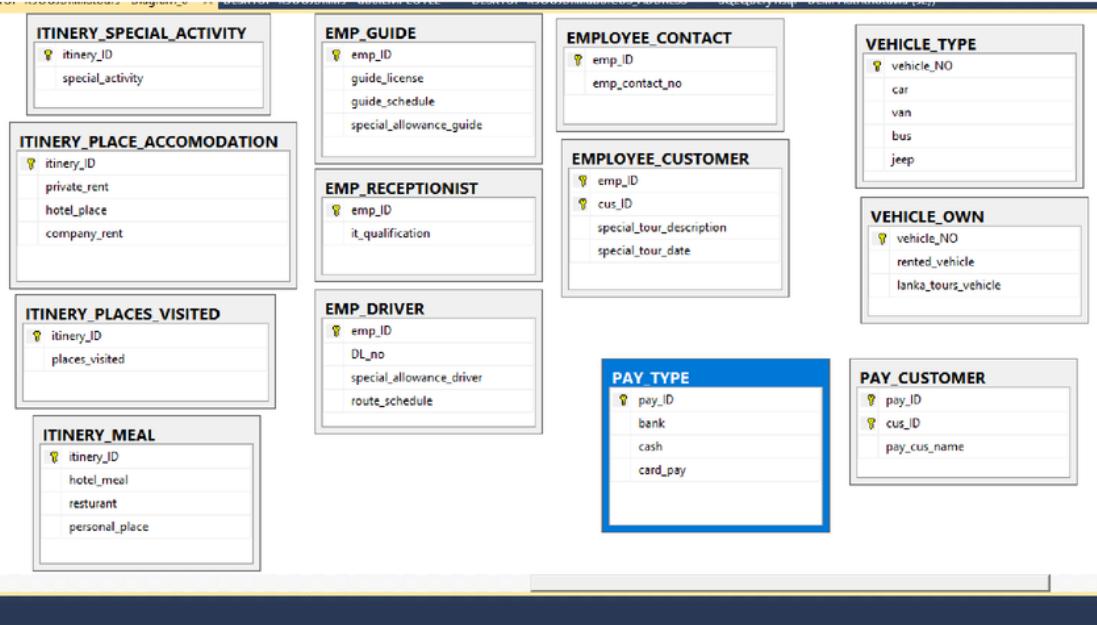
ADD FOREIGN KEY (tour\_ID) REFERENCES TOUR(tour\_ID);

ALTER TABLE RESERVATION

ADD FOREIGN KEY (cus\_ID) REFERENCES CUSTOMER(cus\_ID);

## Database Diagram





## Table with sample records

### CUSTOMER Table

The screenshot shows the MySQL Workbench interface with the CUSTOMER table results tab selected. The table has columns: cus\_ID, cus\_name, gender, nationality, email, fax\_no, and contact\_no. The data consists of 10 rows of customer information.

	cus_ID	cus_name	gender	nationality	email	fax_no	contact_no
1	1	nethmi	F	sri lankan	nethmi@gmail.com	112745063	718762705
2	2	anuja	F	sri lankan	anuja45@gmail.com	112745063	718762705
3	3	jerry	M	American	jerry97@gmail.com	112587983	752397463
4	4	bimsara	M	sri lankan	bimba@gmail.com	114523984	782234567
5	5	malith	M	srlankan	malith123@gmail.com	112387345	752315673
6	6	ravini	F	indian	ravini45@gmail.com	116734965	732415674
7	7	jack	M	ethiyopian	jack34@gmail.com	118723451	721039672
8	8	jill	F	indian	jill56@gmail.com	1135672386	239816527
9	9	amila	M	sri lankan	amila21@gmail.com	112763985	712308356
10	10	yasiru	M	indian	yasi23@gmail.com	118234507	751034562

Query executed successfully.

Ready

### CUS\_ADDRESS Table

The screenshot shows the MySQL Workbench interface with the CUS\_ADDRESS table results tab selected. The table has columns: cus\_ID, post\_box\_no, street, city, province, and country. The data consists of 10 rows of address information.

	cus_ID	post_box_no	street	city	province	country
1	1	159	Gammana road	Maharagama	Western	Sri lanka
2	2	76	Temple road	Pannipitiya	Westem	Sri lanka
3	3	678	Richmond street	Chicago	NULL	USA
4	4	21	Old kottawa road	Kottawa	NULL	Sri lanka
5	5	65	Sunila road	Navinna	NULL	Sri lanka
6	6	2345	Ramadaan road	Kolkata	Maharashtra	India
7	7	1004	Serif street	NULL	NULL	Australia
8	8	456	Periyakulan road	T nagar	Chennai	India
9	9	34	Sisila pedesa	Pannipitiya	Western	Sri lanka
10	10	522	NULL	NULL	NULL	India

Query executed successfully.

Ready

### CUS\_ID\_TYPE Table

Full-screen Snip

100 %

Results Messages

	cus_ID	NIC	DL_no	passport
1	1	977530083V	NULL	NULL
2	2	717402995V	NULL	NULL
3	3	NULL	NULL	D5672935
4	4	NULL	NULL	NULL
5	5	953295623V	NULL	NULL
6	6	NULL	NULL	N3492573
7	7	NULL	NULL	S6723054
8	8	NULL	NULL	N3492710
9	9	973295781V	NULL	NULL
10	10	NULL	NULL	N3599230

Query executed successfully.

Ready

### CUS\_STATUS Table

100 %

Results Messages

	cus_ID	old_cus	current_cus
1	1	yes	
2	2	NULL	yes
3	3	NULL	NULL
4	4	yes	NULL
5	5	yes	NULL
6	6	NULL	yes
7	7	NULL	yes
8	8	yes	NULL
9	9	yes	NULL
10	10	yes	NULL

Query executed successfully.

Ready

## EMPLOYEE Table

100 %

Results Messages

	emp_ID	emp_name	gender	emp_address	salary	cus_ID
1	2000	Keerthi perera	M	no26,depanama,maharagama.	40000	1
2	2001	Kushan cabraal	M	no159/26c,temple road,koswatta.	60000	2
3	2002	Taniya fernando	F	no34,pamunuwa road,maharagama	50000	3
4	2004	Asanka de silva	M	no90,dehiwala road,katuwana.	40000	4
5	2005	Navod kumara	M	no23/6,sumudu road,boralasgamuwa.	45000	5
6	2006	Sumudu perera	F	no67,jayasekara road,nugegoda	50000	6
7	2007	Githmi kaushalya	F	no12,vijeraama,nugegoda.	60000	7
8	2008	Saliya perera	M	no65,silva maawatha,kirulapona.	70000	8
9	2009	Duli gunawardene	F	no167/b,palawatta,baththaramulla.	40000	9
10	2010	Nuwan rathnayaka	M	no89,dewala road,pollwatththa.	45000	5
11	2011	Nathasha kaumadi	F	no56/a,sunila maawatha,homagama	50000	6

Query executed successfully.

Ready

## EMP\_DRIVER Table

100 %

Results Messages

	emp_ID	DL_no	special_allowance_driver	route_schedule
1	2004	B8765	6500	from kandy to galle
2	2005	B0099	20000	from pitipana to anuradhapura
3	2007	B5678	9000	from colombo to trincomalee
4	2008	B5678	5000	from kandy to jaffna
5	2009	B3344	8000	from colombo to meemure
6	2010	B1234	12000	from colombo to galle
7	2011	B1122	5000	from colombo to chilaw
8	2012	B5678	3000	from galle to riverston
9	2015	B3409	8500	from colombo to hatton
10	2018	B3465	6000	from colombo to hanthana

Query executed successfully.

Ready

### **EMP\_GUIDE Table**

The screenshot shows a database query results window with a title bar "100 %". Below it are two tabs: "Results" (selected) and "Messages". The main area displays a table with four columns: emp\_ID, guide\_license, guide\_schedule, and special\_allowance\_guide. The data is as follows:

	emp_ID	guide_license	guide_schedule	special_allowance_guide
1	2000	60007N	no tours on this weekend	NULL
2	2001	60008N	two day tours on this week	10000
3	2004	60009N	no tours on this week	20000
4	2005	60010N	one day tour on tomorrow	10000
5	2007	60020N	weekend trip on next month first week	15000
6	2011	60025N	oneday trip on tomorrow	6700
7	2013	60023N	no tours on this month	NULL
8	2015	60012N	twoday tour on tomorrow	NULL
9	2016	60014N	no tours on this week	3500
10	2018	60054N	no tours on this month	NULL

At the bottom left of the results area, there is a green checkmark icon followed by the text "Query executed successfully." To the right of the results area is a dark blue bar containing the word "Ready".

### **EMP\_RECEPTIONIST Table**

The screenshot shows a database query results window with a title bar "100 %". Below it are two tabs: "Results" (selected) and "Messages". The main area displays a table with two columns: emp\_ID and it\_qualification. The data is as follows:

	emp_ID	it_qualification
1	2002	BSc(honors)IT skills
2	2006	Microsoft word skills
3	2007	Multimedia course
4	2012	Skills in microsoft word
5	2013	BSc(honors)computer hardware
6	2015	Multimedia course
7	2017	BSc(honors)IT skills

At the bottom left of the results area, there is a green checkmark icon followed by the text "Query executed successfully." To the right of the results area is a dark blue bar containing the word "Ready".

## **EMPLOYEE\_CUSTOMER**

The screenshot shows a database query results window with the following details:

- Results Tab:** Selected tab.
- Messages Tab:** Unselected tab.
- Table Headers:** emp\_ID, cus\_ID, special\_tour\_description, special\_tour\_date.
- Data Rows:** 10 rows of data.
- Message Bar:** Shows "Query executed successfully."
- Status Bar:** Shows "Ready".

	emp_ID	cus_ID	special_tour_description	special_tour_date
1	2000	11	oneday tour-galle	2011-02-12
2	2000	12	oneday tour-meemure	2017-02-14
3	2001	13	oneday tour-galle	2017-06-12
4	2002	14	threedays tour-badulla	2017-08-25
5	2005	19	oneday tour-kithulgala	2016-01-14
6	2005	20	twoday tour-ella	2016-02-05
7	2007	15	twodays tour-jaffna	2016-09-23
8	2008	16	weekend tour-anuradapura	2013-05-06
9	2011	17	oneday tour-galle	2015-09-23
10	2017	18	twoday tour-chilaw	2015-11-07

## **EMPLOYEE\_CONTACT**

The screenshot shows a database query results window with the following details:

- Results Tab:** Selected tab.
- Messages Tab:** Unselected tab.
- Table Headers:** emp\_ID, emp\_contact\_no.
- Data Rows:** 11 rows of data.
- Message Bar:** Shows "Query executed successfully."
- Status Bar:** Shows "Ready".

	emp_ID	emp_contact_no
1	2000	712534865
2	2001	781236723
3	2002	723419876
4	2003	712800634
5	2004	775123345
6	2005	773459123
7	2006	716655219
8	2007	782312789
9	2008	772419763
10	2009	723411200
11	2010	772481264

## ITINERY Table

The screenshot shows a SQL query results window with the following details:

- Toolbar: Includes a zoom icon (100%), a refresh icon, and tabs for "Results" and "Messages".
- Table: The "Results" tab displays the data from the ITINERY table. The columns are "itinerary\_ID", "tour\_date", and "tour\_ID". The data consists of 10 rows, each with a value for "itinerary\_ID" (ranging from 1 to 10), "tour\_date" (e.g., 2015-04-26, 2015-09-23, ..., 2016-12-03), and "tour\_ID" (e.g., 4001, 4002, ..., 4010).
- Message Bar: A yellow bar at the bottom indicates "Query executed successfully." with a green checkmark icon.
- Status Bar: The status bar at the bottom of the window says "Ready".

	itinerary_ID	tour_date	tour_ID
1	3001	2015-04-26	4001
2	3002	2015-09-23	4002
3	3003	2016-01-10	4003
4	3004	2016-02-26	4004
5	3005	2016-03-15	4005
6	3006	2016-04-27	4006
7	3007	2016-09-21	4007
8	3008	2016-11-25	4008
9	3009	2016-11-26	4009
10	3010	2016-12-03	4010

## ITINERY\_MEAL Table

The screenshot shows a SQL query results window with the following details:

- Toolbar: Includes a zoom icon (100%), a refresh icon, and tabs for "Results" and "Messages".
- Table: The "Results" tab displays the data from the ITINERY\_MEAL table. The columns are "itinerary\_ID", "hotel\_meal", "restaurant", and "personal\_place". The data consists of 10 rows, each with a value for "itinerary\_ID" (ranging from 1 to 10), "hotel\_meal" (e.g., yes, NULL), "restaurant" (e.g., yes, NULL), and "personal\_place" (e.g., NULL, yes).
- Message Bar: A yellow bar at the bottom indicates "Query executed successfully." with a green checkmark icon.
- Status Bar: The status bar at the bottom of the window says "Ready".

	itinerary_ID	hotel_meal	restaurant	personal_place
1	3001	yes	yes	NULL
2	3002	NULL	yes	yes
3	3003	yes	yes	yes
4	3004	NULL	yes	NULL
5	3005	yes	yes	NULL
6	3006	yes	NULL	yes
7	3007	NULL	yes	yes
8	3008	yes	yes	NULL
9	3009	yes	NULL	yes
10	3010	yes	NULL	NULL

### **ITINERY\_PLACE\_ACCOMODATION Table**

The screenshot shows a SQL query results window with the following details:

- Results Tab:** Selected tab.
- Messages Tab:** Unselected tab.
- Table Structure:**

	itinerary_ID	private_rent	hotel_place	company_rent
1	3001	yes	yes	NULL
2	3002	NULL	yes	yes
3	3003	NULL	NULL	yes
4	3004	NULL	NULL	yes
5	3005	NULL	yes	NULL
6	3006	NULL	yes	NULL
7	3007	NULL	yes	NULL
8	3008	yes	yes	NULL
9	3009	NULL	NULL	yes
10	3010	yes	yes	NULL
- Status Bar:** Shows a green checkmark icon and the message "Query executed successfully."
- Bottom Bar:** Shows the word "Ready".

### **ITINERY\_PLACES\_VISITED Table**

The screenshot shows a SQL query results window with the following details:

- Results Tab:** Selected tab.
- Messages Tab:** Unselected tab.
- Table Structure:**

	itinerary_ID	places_visited
1	3001	hortain plains,hanthana
2	3002	galle fort,hikkaduwa
3	3003	NULL
4	3004	nuckles,ella
5	3005	galle fort,hikkaduwa
6	3006	peradeniya flower garden,teple of the tooth relic
7	3007	NULL
8	3008	museum,kaluthara temple
9	3009	nuckles,ella
10	3010	peradeniya flower garden,teple of the tooth relic
- Status Bar:** Shows a green checkmark icon and the message "Query executed successfully."
- Bottom Bar:** Shows the word "Ready".

### **ITINERY\_SPECIAL\_ACTIVITY Table**

The screenshot shows a SQL query results window with the following details:

- Toolbar:** Includes a zoom dropdown set to 100%, a "Results" tab (selected), and a "Messages" tab.
- Table:** The "Results" tab displays the data from the ITINERY\_SPECIAL\_ACTIVITY table. The columns are "itinerary\_ID" and "special\_activity". The data consists of 10 rows:

	itinerary_ID	special_activity
1	3001	NULL
2	3002	archery,shooting
3	3003	boat riding
4	3004	NULL
5	3005	hiking
6	3006	shooting,watching film
7	3007	achery
8	3008	NULL
9	3009	camping,hiking
10	3010	water rafting,boat riding

- Status Bar:** Shows a green checkmark icon and the message "Query executed successfully."
- Bottom Bar:** Shows the word "Ready".

### **PAYMENT Table**

The screenshot shows a SQL query results window with the following details:

- Toolbar:** Includes a zoom dropdown set to 100%, a "Results" tab (selected), and a "Messages" tab.
- Table:** The "Results" tab displays the data from the PAYMENT table. The columns are "pay\_ID", "total\_pay\_re...", "advance\_pay\_re...", "total\_pay\_d...", "advance\_pay\_d...", and "cus\_ID". The data consists of 10 rows:

	pay_ID	total_pay_re...	advance_pay_re...	total_pay_d...	advance_pay_d...	cus_ID
1	6001	issused	issused	2016-02-25	2016-02-27	1
2	6002	issused	issused	2016-03-25	2016-03-04	2
3	6004	issused	issused	2016-04-12	2016-04-23	3
4	6005	issused	issused	2016-05-04	2016-05-01	4
5	6006	issused	issused	2016-07-08	2016-07-11	5
6	6007	issused	issused	2016-08-08	2016-08-05	6
7	6008	issused	issused	2016-09-09	2016-09-05	7
8	6009	issused	issused	2016-12-23	2016-12-12	8
9	6010	issused	issused	2017-01-03	2016-12-30	9
10	6011	issused	issused	2017-02-10	2017-10-04	10

- Status Bar:** Shows a green checkmark icon and the message "Query executed successfully." on the left, and the computer name "DESKTOP-DONOI" on the right.

### **PAY\_CUSTOMER Table**

The screenshot shows a database management system interface with a results grid. The grid has three columns: pay\_ID, cus..., and pay\_cus\_na... . The data consists of 10 rows, each containing a value for pay\_ID (ranging from 1 to 10), a value for cus... (ranging from 1 to 10), and a value for pay\_cus\_na... (containing names like keerthi, malith, bimsara, etc.).

	pay_ID	cus...	pay_cus_na...
1	6001	1	keerthi
2	6002	5	malith
3	6003	4	bimsara
4	6004	6	ravini
5	6005	9	amila
6	6007	2	anuja
7	6008	3	jerry
8	6009	7	jack
9	6010	10	yasiru

Query executed successfully. | DESKTOP-D0NOI8V (12.0)

### **PAY\_TYPE Table**

The screenshot shows a database management system interface with a results grid. The grid has four columns: pay\_ID, bank, cash, and card\_p... . The data consists of 10 rows, each containing a value for pay\_ID (ranging from 1 to 10), a value for bank (mostly paid, some NULL), a value for cash (mostly NULL, some paid), and a value for card\_p... (mostly NULL, some paid).

	pay_ID	bank	cash	card_p...
1	6001	paid	NULL	NULL
2	6002	NULL	NULL	paid
3	6003	paid	NULL	NULL
4	6004	NULL	paid	NULL
5	6005	paid	NULL	NULL
6	6006	NULL	NULL	paid
7	6007	paid	NULL	NULL
8	6008	NULL	paid	NULL
9	6009	NULL	paid	NULL
10	6010	paid	NULL	NULL

Query executed successfully. | DESKTOP-D0NOI8V (12.0)

## **RESERVATION Table**

The screenshot shows a SQL query results window with the following details:

- Toolbar:** Includes a zoom icon (100%), a back arrow, and a forward arrow.
- Tab Bar:** Shows "Results" and "Messages".
- Table:** A grid displaying 10 rows of data from the RESERVATION table. The columns are labeled: res\_ID, emp\_ID, cus..., and res\_date.
- Status Bar:** Shows a green checkmark icon followed by the text "Query executed successfully." and the computer name "DESKTOP-D0NOI8V (12)".

	res_ID	emp_ID	cus...	res_date
1	5001	2000	1	2015-04-23
2	5002	2008	7	2015-09-12
3	5003	2007	6	2016-01-03
4	5004	2000	5	2016-02-23
5	5006	2011	3	2016-04-25
6	5007	2002	2	2016-08-15
7	5008	2001	9	2018-01-05
8	5009	2008	4	2018-02-23
9	5010	2005	10	2018-02-24
10	5011	2002	4	2018-02-24

## **RESERVATION\_CONTACT Table**

The screenshot shows a SQL query results window with the following details:

- Toolbar:** Includes a zoom icon (100%), a back arrow, and a forward arrow.
- Tab Bar:** Shows "Results" and "Messages".
- Table:** A grid displaying 10 rows of data from the RESERVATION\_CONTACT table. The columns are labeled: res\_ID and res\_cus\_contact...>.
- Status Bar:** Shows a green checkmark icon followed by the text "Query executed successfully." and the computer name "DESKTOP-D0NOI8V (12)".

	res_ID	res_cus_contact...
1	5001	712435098
2	5002	721845023
3	5003	782138753
4	5004	721956734
5	5005	714290074
6	5007	724567145
7	5008	782312987
8	5009	721345298
9	5010	712345617
10	5011	758213456

### **RESERVATION\_CUSTOMER Table**

The screenshot shows a SQL Server Management Studio window with the 'Results' tab selected. The query results are displayed in a table with three columns: res\_ID, cus..., and res\_cus\_na... . The data consists of 10 rows, each containing a value for res\_ID (ranging from 1 to 10), a value for cus... (ranging from 1 to 10), and a name (nethmi, anuja, jerry, bimsara, malith, ravini, jack, jill, amila, yasiru). A message bar at the bottom indicates that the query was executed successfully.

	res_ID	cus...	res_cus_na...
1	5001	1	nethmi
2	5002	2	anuja
3	5003	3	jerry
4	5004	4	bimsara
5	5005	5	malith
6	5006	6	ravini
7	5007	7	jack
8	5008	8	jill
9	5009	9	amila
10	5010	10	yasiru

Query executed successfully. | DESKTOP-D0NOI8

### **RESERVATION\_EMPLOYEE Table**

The screenshot shows a SQL Server Management Studio window with the 'Results' tab selected. The query results are displayed in a table with three columns: res\_ID, emp\_ID, and res\_emp\_name . The data consists of 10 rows, each containing a value for res\_ID (ranging from 1 to 10), a value for emp\_ID (ranging from 2000 to 2009), and a name (keerthi perera, keerthi perera, taniya fernando, taniya fernando, sumudu perera, sumudu perera, sumudu perera, duli gunawardene, duli gunawardene, taniya fernando). A message bar at the bottom indicates that the query was executed successfully.

	res_ID	emp_ID	res_emp_name
1	5001	2000	keerthi perera
2	5002	2000	keerthi perera
3	5003	2002	taniya fernando
4	5004	2002	taniya fernando
5	5005	2006	sumudu perera
6	5006	2006	sumudu perera
7	5007	2006	sumudu perera
8	5009	2009	duli gunawardene
9	5010	2009	duli gunawardene
10	5011	2002	taniya fernando

Query executed successfully. | DESKTOP-D0NOI8V (12)

## TOUR Table

The screenshot shows the MySQL Workbench interface with a query results window. The results tab is selected, displaying the following data:

	tour_ID	tour_start_d...	tour_end_d...	issue_tour_...	res_ID
1	4001	2015-04-26	2015-04-27	issused	5001
2	4002	2015-09-23	2015-09-25	issused	5002
3	4003	2016-01-10	2016-01-14	issused	5003
4	4004	2016-02-26	2016-02-27	issused	5004
5	4005	2016-03-15	2016-03-20	issused	5004
6	4006	2016-04-27	2016-04-30	issused	5006
7	4007	2016-09-21	2016-09-24	issused	5007
8	4008	2016-11-25	2016-11-24	issused	5008
9	4009	2016-11-26	2016-11-29	issused	5009
10	4010	2016-12-03	2016-12-05	issused	5010

At the bottom of the results window, a green checkmark icon and the text "Query executed successfully." are visible. To the right, the computer name "DESKTOP-D0NOI8V" is displayed.

## TOUR\_LOCATION Table

The screenshot shows the MySQL Workbench interface with a query results window. The results tab is selected, displaying the following data:

	tour_ID	location_pickup_participant	location_drop_participant
1	4000	lankatours	kottawa pizza hut
2	4002	nugegoda pizza hut	homagama
3	4003	maharagama bus stand	maharagama bus stand
4	4004	katunayaka airport	katunayaka airport
5	4005	no45,sri nayaka road,palawaththa	katunayaka airport
6	4006	kottawa bus stand	kottawa bus stand
7	4007	hilton hotel	katunayaka air port
8	4008	no34,dewin road,pannipitiya	no34,dewin road,pannipitiya
9	4009	kottawa KFC	horana town
10	4010	nugegoda	green hotel,colombo

At the bottom of the results window, a green checkmark icon and the text "Query executed successfully." are visible. To the right, the computer name "DESKTOP-D0NOI8V (12.0)" is displayed.

## TOUR\_PARTICIPANT Table

100 % <

Results Messages

	tour_ID	list_of_participant
1	4001	meera fernando,nipun sandeepa,asanka sanath,anja...
2	4002	nethmmi divya,amila nipun,asela perera,navod kuma...
3	4003	bimsara perera,malith boralogoda,aloka perera,malis...
4	4004	aliya batt,deepika padukone,teena,fernando,samana...
5	4005	kevin nugera,saliya perera,nuwangi silva
6	4006	tharindu shehan,fauzaan ashir.tehan perera,denuwa...
7	4007	malka kaushi,kavi weerasinghe,suneth gamage,vihari...
8	4008	ravini methma,thisari de silva
9	4009	sandun dananjaya,amila silva,akila perera
10	4010	dinithi silva,dimithra bandara,deshan perera

Query executed successfully. DESKTOP-DONC

## TOUR\_TYPE Table

100 % <

Results Messages

	tour_ID	lanka_tours_...	guest_tour_...
1	4001	made	NULL
2	4002	NULL	made
3	4003	NULL	made
4	4004	made	NULL
5	4005	made	NULL
6	4006	NULL	made
7	4007	NULL	made
8	4008	NULL	made
9	4009	made	NULL
10	4010	NULL	made

Query executed successfully. DESKTOP-DONC

## TOUR\_STATUS Table

The screenshot shows a SQL query results window with the following details:

- Toolbar:** Includes a zoom icon (100%), a back arrow, and a forward arrow.
- Tab Bar:** Shows "Results" and "Messages".
- Table:** A grid displaying 10 rows of data from the TOUR\_STATUS table. The columns are: tour\_ID, in\_fut..., in\_progr..., and complete. The data is as follows:

	tour_ID	in_fut...	in_progr...	complete
1	4001	NULL	NULL	completed
2	4002	NULL	NULL	completed
3	4003	NULL	NULL	completed
4	4004	NULL	NULL	completed
5	4005	NULL	inprogress	NULL
6	4006	future	NULL	NULL
7	4007	future	NULL	NULL
8	4008	future	NULL	NULL
9	4009	NULL	inprogress	NULL
10	4010	NULL	inprogress	NULL

**Status Bar:** Displays a green checkmark icon and the message "Query executed successfully." followed by the computer name "DESKTOP-D0NOI8".

## VEHICLE Table

The screenshot shows a SQL query results window with the following details:

- Toolbar:** Includes a zoom icon (100%), a back arrow, and a forward arrow.
- Tab Bar:** Shows "Results" and "Messages".
- Table:** A grid displaying 10 rows of data from the VEHICLE table. The columns are: vehicle..., tour\_dista..., route\_description, rate\_per\_..., and tour\_ID. The data is as follows:

	vehicle...	tour_dista...	route_descripti...	rate_per_...	tour_ID
1	8001	560km	colombo to kandy	300.00	4001
2	8002	344km	colombo to galle	650.00	4002
3	8003	670km	colombo to trincomalee	230.00	4003
4	8004	345km	chilaw to galle	150.00	4004
5	8005	890k	colobo to anradhapura	400.00	4005
6	8006	1024km	colombo to jaffna	300.00	4006
7	8007	2700km	anuradhapura to galle	250.00	4007
8	8008	596km	colombo to ella	300.00	4008
9	8009	780km	kottawa to badulla	150.00	4009
10	8010	678km	dabulla to colombo	230.00	4010

**Status Bar:** Displays a green checkmark icon and the message "Query executed successfully." followed by the computer name "DESKTOP-D0NOI8".

### **VEHICLE\_OWN Table**

The screenshot shows a database management system interface with a results grid. The table has three columns: vehicle\_id, rented\_vehicle, and lanka\_tours\_vehicle. The data consists of nine rows, each containing a vehicle ID and either 'yes' or 'NULL' for the other two fields.

	vehicle_id	rented_vehicle	lanka_tours_vehicle
1	8001	NULL	yes
2	8002	NULL	yes
3	8003	yes	NULL
4	8004	NULL	yes
5	8005	NULL	yes
6	8006	yes	NULL
7	8007	yes	NULL
8	8008	yes	NULL
9	8009	NULL	yes

Query executed successfully. | DESKTOP-D0NOI8V (12.0 S)

### **VEHICLE\_TYPE Table**

The screenshot shows a database management system interface with a results grid. The table has six columns: vehicle\_id, car, van, bus, jeep, and truck. The data consists of ten rows, each containing a vehicle ID and either 'yes' or 'NULL' for the other five fields.

	vehicle_id	car	van	bus	jeep	truck
1	8001	NULL	NULL	yes	NULL	NULL
2	8002	NULL	yes	NULL	yes	NULL
3	8003	NULL	NULL	NULL	yes	NULL
4	8004	yes	yes	NULL	NULL	NULL
5	8005	NULL	NULL	yes	NULL	NULL
6	8006	yes	NULL	NULL	yes	NULL
7	8007	yes	NULL	NULL	NULL	NULL
8	8008	NULL	yes	yes	NULL	NULL
9	8009	NULL	NULL	yes	NULL	NULL
10	8010	NULL	yes	NULL	NULL	NULL

Query executed successfully. | DESKTOP-D0NOI8V (12.0 S)

## Create Triggers

### **3 INSERT**

```
CREATE TRIGGER Empinsert
ON [EMPLOYEE]
INSTEAD OF INSERT
AS
begin
declare @empid bigint;
declare @empname varchar(255);
declare @empgender varchar(1);
declare @empaddress varchar(255);

declare @empsal int;
3
select @empid = i.emp_ID from inserted i;

select @empname = i.emp_name from inserted i;
select @empgender = i.gender from inserted i;
select @empaddress = i.emp_address inserted i;
select @empsal = i.salary inserted i;
1
INSERT INTO EMPLOYEE
(emp_ID , emp_name , gender , emp_address , salary)
VALUES( @empid , @empname , @empgender , @empaddress , @empsal);
GO
```

Object Explorer    SQLQuery1.sql - DE...\\Haththotuwa (52) \* X

```
CREATE TRIGGER Empinsert
ON [EMPLOYEE]
INSTEAD OF INSERT
AS
begin
declare @empid bigint;
declare @empname varchar(255);
declare @empgender varchar(1);
declare @empaddress varchar(255);
declare @empsal int;

select @empid=i.emp_ID from inserted i;
select @empname=i.emp_name from inserted i;
select @empgender=i.gender from inserted i;
select @empaddress=i.emp_address from inserted i;
select @empsal=i.salary from inserted i;
INSERT INTO EMPLOYEE
```

100 %

Messages

Command(s) completed successfully.

100 %

Query executed successfully.

DESKTOP-K90OJD1 (11.0 RTM) | DESKTOP-K90OJD1

Ready

Ln 24 Col 1

## UPDATE

```
CREATE TRIGGER empupdate
ON [EMPLOYEE]
3
AFTER INSERT
AS
declare @empid bigint;
declare @empname varchar(255);
declare @empgender varchar(1);
declare @empaddress varchar(255);
declare @empsal int;
1
declare @audit_action varchar(255);

select @empid = i.emp_ID from inserted i;
select @empname = i.emp_name from inserted i;
select @empsal = i.salary inserted i;
if update ( emp_name )
    set @audit_action = 'Updated Record—After Update Trigger';
if update( salary )
    set @audit_action = 'Updated Record—After Update Trigger';

INSERT INTO EMPLOYEE
(emp_ID , emp_name , gender , emp_address , salary)
VALUES( @empid , @empname , @empgender , @empaddress , @empsal);
PRINT 'AFTER INSERT trigger fired' ;
GO
```

Object Explorer    SQLQuery1.sql - DE..il Haththotuwa (52) \*

```
CREATE TRIGGER empupdate
ON [EMPLOYEE]
AFTER INSERT
AS
declare @empid bigint;
declare @empname varchar(255);
declare @empgender varchar(1);
declare @empaddress varchar(255);
declare @empsal int;
declare @audit_action varchar(255);

select @empid=i.emp_ID from inserted i;
select @empname=i.emp_name from inserted i;
select @empgender=i.gender from inserted i;
select @empaddress=i.emp_address from inserted i;
select @empsal=i.salary from inserted i;
```

100 %

Messages

Command(s) completed successfully.

100 %

Query executed successfully.

DESKTOP-K90OJDI (11.0 RTM) DESKTOP-K90OJD\Githm... lk

Matches: BEGIN

Ln 52    Col 3

## TRIGGER EXECUTION

UPDATE EMPLOYEE

SET salary =40000

WHERE emp\_ID = 2005

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a connection to 'DE...i Haththotuwa (S2)' is selected. In the center pane, a query window displays the following T-SQL code:

```
Object Explorer   SQLQuery1.sql - DE...i Haththotuwa (S2)* ×
set @audit_action='Updated record->After update Trigger';

INSERT INTO EMPLOYEE
(emp_ID,emp_name,gender,emp_address,salary) VALUES
(@empid,@epname,@empgender,@empaddress,@empsal);
print 'AFTER UPDATE TRIGGER fired'
GO

update EMPLOYEE
SET salary=40000
where emp_ID=2005
```

The code consists of two parts: an `INSERT` statement that triggers a print statement and a `GO` keyword, followed by an `update` statement that changes the salary of employee ID 2005. Below the code, the 'Messages' tab shows the output: '(1 row(s) affected)'. At the bottom of the screen, a status bar indicates 'Query executed successfully.' and shows the system name 'DESKTOP-K900JDI (11.0 RTM) | DESKTOP-K900JDI'.

## DELETE

```
CREATE TRIGGER empdelete
ON [EMPLOYEE]
AFTER DELETE
3
AS
declare @empid bigint;
declare @empname varchar(255);
declare @empgender varchar(1);
declare @empaddress varchar(255);
declare @empsal int;
1
declare @audit_action varchar(255);

select @empid = d.emp_ID from deleted i;
select @empname = d.emp_name from deleted i;
select @empgender = d.gender from deleted i;
select @empaddress = d.emp_address deleted i;
select @empsal = d.salary deleted i;
```

The screenshot shows the Object Explorer and a SQL Query window. The query window displays the T-SQL code for creating a trigger named 'empdelete' on the 'EMPLOYEE' table, which fires after a delete operation. The trigger declares variables for employee ID, name, gender, address, and salary, and then selects these values from the deleted table. A status bar at the bottom indicates 'Query executed successfully.'

```
Object Explorer   SQLQuery1.sql - DE_Employee (52)  X
CREATE TRIGGER empdelete
ON [EMPLOYEE]
AFTER DELETE
AS
declare @empid bigint;
declare @empname varchar(255);
declare @empgender varchar(1);
declare @empaddress varchar(255);
declare @empsal int;
declare @audit_action varchar(255);

select @empid=d.emp_ID from deleted d;
select @empname=d.emp_name from deleted d;
select @empgender=d.gender from deleted d;
select @empaddress=d.emp_address from deleted d;
select @empsal=d.salary from deleted d;
select @audit_action='Deleted--After Delete Trigger.';

100 %
Messages
Command(s) completed successfully.

100 % < 
Query executed successfully.  DESKTOP-K300JDI (11.0 RTM)  DESKTOP-K300JDI
Matches: BEGIN  Ln 79  Col 3
```

## Create Functions Statements

```
CREATE FUNCTION dbo.TourVehicleType
( @tour_ID bigint , @vehicle_type varchar(45) , @van varchar(45))
RETURNS
TABLE
AS
RETURN
( SELECT tour_ID , vehicle_NO , van
FROM TOUR T , VEHICLE_TYPE V
WHERE tour_ID = @tour_ID and VAN = @van
```

The screenshot shows the SQL Query Editor window in SQL Server Management Studio. The code in the query pane is:

```
CREATE FUNCTION dbo.TourVehicleType
( @tour_ID bigint , @vehicle_type varchar(45) , @van varchar(45))
RETURNS
TABLE
AS
RETURN
( SELECT tour_ID , vehicle_NO , van
FROM TOUR T , VEHICLE_TYPE V
WHERE tour_ID = @tour_ID and VAN = @van )
```

The code is highlighted with syntax coloring. Below the editor, the Messages pane displays:

Commands completed successfully.

At the bottom of the screen, the status bar shows:

100 %    100 %    ✓ Query executed successfully. DESKTOP-1FMAJR4\SQLEXPRESS ... | DESKTOP-1FMAJR4\Shenal... | lankatours | 00:00:00 | 0

## FUNCTION EXECUTION

SELECT \* FROM TOUR , VEHICLE\_TYPE WHERE tour\_ID LIKE 4002 AND VAN LIKE van

The screenshot shows a SQL Server Management Studio window. The top pane displays a script for creating a function named 'TourVehicleType'. The function takes two parameters: 'tour\_ID' (bigint) and 'van' (varchar(45)). It returns a table with columns: tour\_ID, tour\_start\_date, tour\_end\_date, issue\_tour\_map, res\_ID, vehicle\_NO, car, van, bus, and jeep. The body of the function uses a SELECT statement to query the 'TOUR' and 'VEHICLE\_TYPE' tables, filtering by tour\_ID and van. The bottom pane shows the results of executing this function with tour\_ID 4002 and van 'van'. The results are as follows:

	tour_ID	tour_start_date	tour_end_date	issue_tour_map	res_ID	vehicle_NO	car	van	bus	jeep
1	4002	2015-09-23	2015-09-25	issued	5002	8002	NULL	yes	NULL	yes
2	4002	2015-09-23	2015-09-25	issued	5002	8004	yes	yes	NULL	NULL
3	4002	2015-09-23	2015-09-25	issued	5002	8008	NULL	yes	yes	NULL
4	4002	2015-09-23	2015-09-25	issued	5002	8010	NULL	yes	NULL	NULL

Below the results, a message bar indicates "Query executed successfully." and shows connection details: DESKTOP-1FMAJR4\SQLEXPRESS ... | DESKTOP-1FMAJR4\Shenal... | lankatours | 00:00:00 | 4 row.

## Create View Statements

2  
CREATE VIEW CUSTOMER\_TEST  
AS  
SELECT cus\_ID , cus\_name  
FROM CUSTOMER  
WHERE cus\_ID = cus\_ID

### **EXECUTION**

SELECT \* FROM [CUSTOMER\_TEST]

The screenshot shows the SQL Server Management Studio (SSMS) interface. In the top pane, a query window titled 'SQLQuery1.sql - DE...al Megawarna (\$3)\*' contains the following T-SQL code:

```
use lankatours;
CREATE VIEW CUSTOMER_TEST
AS
SELECT cus_ID , cus_name
FROM CUSTOMER
WHERE cus_ID = cus_ID
SELECT * FROM [CUSTOMER_TEST];
```

In the bottom pane, the 'Results' tab displays the output of the executed query. The results are a table with two columns: 'cus\_ID' and 'cus\_name'. The data is as follows:

	cus_ID	cus_name
1	1	neethni
2	2	anuja
3	3	jerry
4	4	bimara
5	5	malith
6	6	ravini
7	7	jack
8	8	jill
9	9	amila
10	10	yasini

A status bar at the bottom indicates: 'Query executed successfully.' and 'DESKTOP-1FMAJR4\SQLEXPRESS ... DESKTOP-1FMAJR4\Shenal... lankatours 00:00:00 | 10 rows'.

```
CREATE VIEW [CUSTOMER_STREET]
AS
SELECT cus_ID , street
FROM CUS_ADDRESS
WHERE street = 'maharagama'
```

## EXECUTION

```
SELECT * FROM [CUSTOMER_STREET]
```

The screenshot shows the SQL Server Management Studio (SSMS) interface. The query window title is "SQLQuery1.sql - DE...al Megawama (53)". The query itself is:

```
CREATE VIEW [CUSTOMER_STREET]
AS
SELECT cus_ID , street
FROM CUS_ADDRESS
WHERE street = 'maharagama'

SELECT*FROM[CUSTOMER_STREET];
```

The results pane shows a table with two columns: "cus\_ID" and "street". There are no rows present.

At the bottom of the results pane, a status bar displays: "Query executed successfully." and "DESKTOP-1FMAJR4\SQLEXPRESS...".

```
CREATE VIEW EMPLOYEE_TYPE
```

```
AS
```

```
SELECT emp_ID , emp_name , gender
```

```
1
```

```
FROM EMPLOYEE
```

```
WHERE emp_ID = emp_ID
```

The screenshot shows a SQL Server Management Studio window titled "SQLQuery1.sql - DE...al Megawarna (33)\*". The query pane contains the T-SQL code for creating a view:

```
CREATE VIEW EMPLOYEE_TYPE
AS
SELECT emp_ID, emp_name, gender
FROM EMPLOYEE
WHERE emp_ID=emp_ID
```

The execution results pane shows the message "Commands completed successfully." and the status bar at the bottom indicates "Query executed successfully." and "0 rows".

## EXECUTION

```
SELECT * FROM [EMPLOYEE_TYPE]
```

The screenshot shows the SQL Server Management Studio interface. A query window titled "SQLQuery1.sql - DE...al Megawarna (53)\*" is open. It contains the following SQL code:

```
CREATE VIEW EMPLOYEE_TYPE
AS
SELECT emp_ID,emp_name,gender
FROM EMPLOYEE
WHERE emp_ID=emp_ID
SELECT * FROM[EMPLOYEE_TYPE]
```

Below the code, there is a "Results" tab showing the output of the query. The results are presented in a table with three columns: emp\_ID, emp\_name, and gender. The data consists of 11 rows:

emp_ID	emp_name	gender
1 2000	Keethi perera	M
2 2001	Kushan cabral	M
3 2002	Tanya fernando	F
4 2004	Aarika de silva	M
5 2005	Navod kumara	M
6 2006	Sumudu perera	F
7 2007	Githmi kaushalya	F
8 2008	Saliya perera	M
9 2009	Duli gunawardene	F
10 2010	Nuwan Rathnayaka	M
11 2011	Nathasha kaumadi	F

At the bottom of the results pane, a green checkmark icon indicates "Query executed successfully." To the right of the results pane, status information is displayed: DESKTOP-1FMAJR4\SQLEXPRESS..., DESKTOP-1FMAJR4\Shenal..., lankatours, 00:00:00 | 19 rows.

The screenshot shows the SQL Server Management Studio interface. A query window titled "SQLQuery1.sql - DE...al Megawarna (53)\*" is open. It contains the same SQL code as the previous screenshot:

```
CREATE VIEW EMPLOYEE_TYPE
AS
SELECT emp_ID,emp_name,gender
FROM EMPLOYEE
WHERE emp_ID=emp_ID
SELECT * FROM[EMPLOYEE_TYPE]
```

Below the code, there is a "Results" tab showing the output of the query. The results are presented in a table with three columns: emp\_ID, emp\_name, and gender. The data consists of 11 rows:

emp_ID	emp_name	gender
1 2000	Keethi perera	M
2 2001	Kushan cabral	M
3 2002	Tanya fernando	F
4 2004	Aarika de silva	M
5 2005	Navod kumara	M
6 2006	Sumudu perera	F
7 2007	Githmi kaushalya	F
8 2008	Saliya perera	M
9 2009	Duli gunawardene	F
10 2010	Nuwan Rathnayaka	M
11 2011	Nathasha kaumadi	F

At the bottom of the results pane, a green checkmark icon indicates "Query executed successfully." To the right of the results pane, status information is displayed: DESKTOP-1FMAJR4\SQLEXPRESS..., DESKTOP-1FMAJR4\Shenal..., lankatours, 00:00:00 | 19 rows.

## Create Procedure Statements

GO

```
CREATE PROCEDURE ItineraryInfo @itinerary_no varchar(45)
```

AS

SELECT\*

FROM ITINERY

```
WHERE itinerary_ID = @itinerary_no
```

)

(

SELECT\*

FROM ITINERY\_PLACES\_VISITED

```
WHERE itinerary_ID = @itinerary_no
```

)

GO

The screenshot shows the SQL Server Management Studio interface. On the left, there are four tabs: 'SQLQuery17.sql ...D0NOI8V\user (70)', 'SQLQuery16.sql ...D0NOI8V\user (69)\*' (which is the active tab), 'SQLQuery15.sql ...D0NOI8V\user (68)', and 'SQLQuery14.sql ...D0NOI8V\user (67)'. The query window contains the T-SQL code for creating the 'ItineraryInfo' stored procedure. The code includes a 'GO' statement, a 'CREATE PROCEDURE' block with parameters, nested 'SELECT' statements from the 'ITINERY' and 'ITINERY\_PLACES\_VISITED' tables, and another 'SELECT' statement within the same scope. The 'Messages' pane at the bottom shows a single message: 'Command(s) completed successfully.' The status bar at the bottom right indicates the connection details: 'DESKTOP-D0NOI8V (12.0 SP1) | DESKTOP-D0NOI8V\user (69) | lankatours | 00:00:00 | 0 rows'. On the right side of the screen, there is a 'Template Browser' window showing a tree view of 'SQL Server Templates' including Aggregate, Assembly, Audit, Backup, Certificate, Change Data Capture, Change Tracking, Credential, Database, Database Mail, Database Role, Database Trigger, Default, Earlier Versions, and EnvPrint. Below the template browser is a 'Properties' window for the current connection, showing details like 'Aggregate Status', 'Connection failure', 'Elapsed time', 'Finish time', 'Name', 'Rows returned', 'Start time', 'State', and 'Connection name'. The 'Name' field in the properties window is set to 'DESKTOP-D0NO'.

```
use lankatours;
GO
CREATE PROCEDURE ItineraryInfo @itinerary_no varchar(45)
AS
(
    SELECT*
    FROM ITINERY
    WHERE itinerary_ID = @itinerary_no
)

(
    SELECT*
    FROM ITINERY_PLACES_VISITED
    WHERE itinerary_ID = @itinerary_no
)
GO
```

100 % < Messages  
Command(s) completed successfully.

100 % < DESKTOP-D0NOI8V (12.0 SP1) | DESKTOP-D0NOI8V\user (69) | lankatours | 00:00:00 | 0 rows

Query executed successfully.

Template Browser

Properties

Aggregate Status

Connection failure

Elapsed time 00:00:00.442

Finish time 2017-12-31 5:40

Name DESKTOP-D0NO

Rows returned 0

Start time 2017-12-31 5:40

State Open

Connection

Connection name DESKTOP-D0NO

Connection Details

Connection elapse 00:00:00.442

Name

## EXECUTION PROCEDURE

EXEC ItineraryInfo @itinerary\_no = 3001

The screenshot shows the SQL Server Management Studio interface. In the center, there is a query editor window with the following T-SQL code:

```
SELECT *  
FROM ITINERARY_PLACES_VISITED  
WHERE itinerary_ID = @itinerary_no  
GO  
  
EXEC ItineraryInfo @itinerary_no = 3001
```

Below the code, the results pane displays two tables. The first table, 'itinerary\_no', has one row with values: itinerary\_no = 3001, tour\_date = 2015-04-26, tour\_ID = 4001. The second table, 'places\_visited', has one row with values: itinerary\_no = 3001, places\_visited = hortain plains, hanthana.

In the bottom status bar, it says "Query executed successfully." and provides connection details: DESKTOP-D0NOIBV (12.0 SP1) | DESKTOP-D0NOIBV\user (69) | lankatours | 00:00:00 | 2 rows.

On the right side of the screen, there is a "Template Browser" window showing a list of SQL Server Templates, and a "Properties" window showing connection parameters and aggregate status.

## **Critical Appraisal**

- The main feature of the database is that the integrity of the data we have used for the details. There we have managed to come with actual tour management system situation data.
- Most of the fields in the tables are applied with triggers and the users of the system can manipulate or deal with the database with less number of errors.
- In some cases, user might find it difficult to insert data for some tables as they are applied with triggers. When a trigger is fired sometime errors to the ongoing process can occur, as an example interruptions and system crashes.

## **Futher Implementation**

- We will connect this database to real world tour management systems and plan develop separate software to this database.
- From the data retrieval, insertion and other manipulations can be done in a user-friendly way.
- After creating the user interfaces for the database, every person can easily deal with this database.
- Advanced triggers added to the database , therefore it reduce the errors that can be happen from the users when inserting and manipulating data.
- After creating the database, can reduce too large space decreasing to many free spaces in the space allocated for the database.

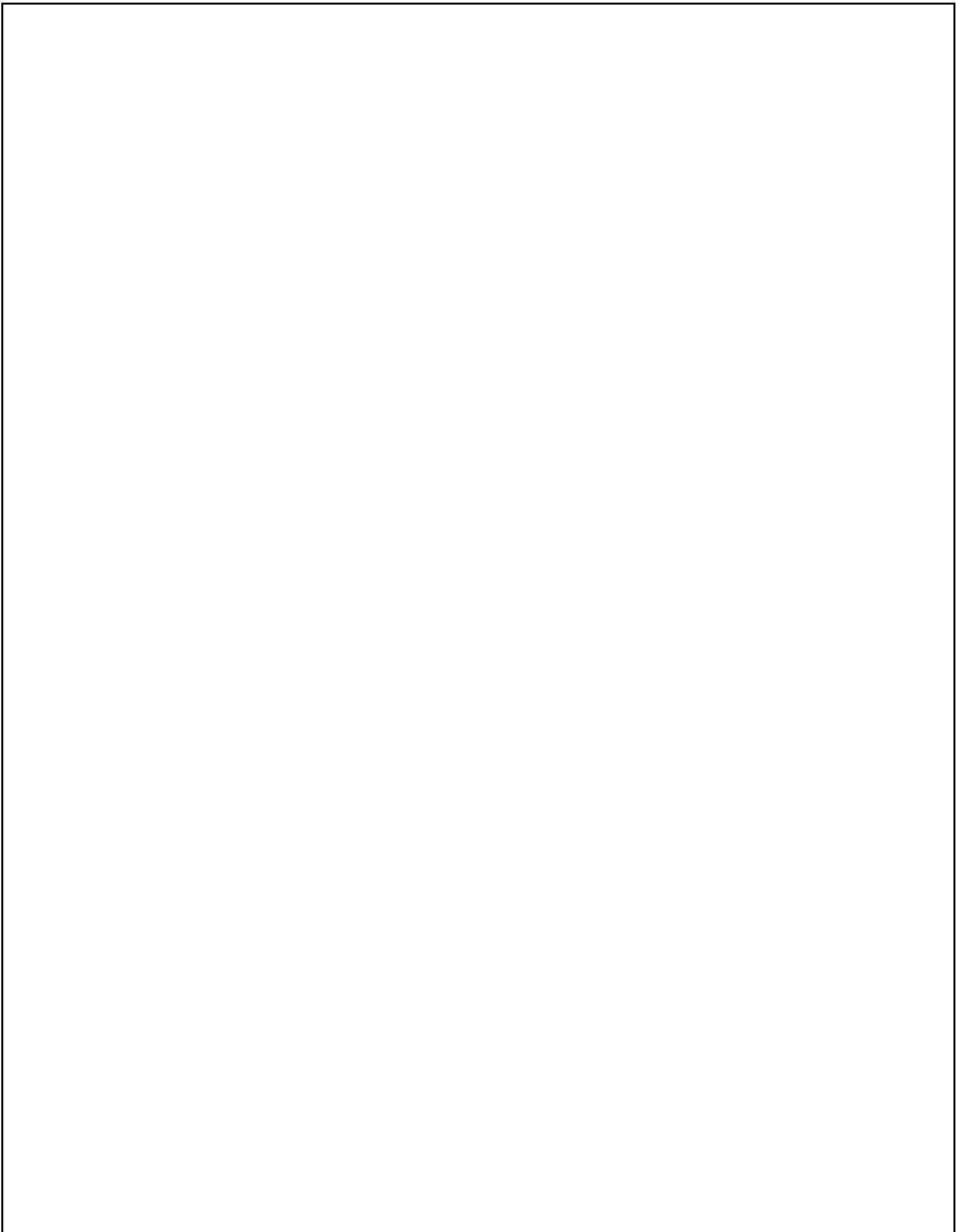
## **Work Load Matrix**

<b>Index Number</b>	<b>ER/EER Diagram</b>	<b>Relational Mapping, Data Normalization, Data Dictionary</b>	<b>Tables, Constraints</b>	<b>Views, Triggers</b>	<b>Stored Procedures, User Defined Functions</b>
10602218	✗	✗			
10602194	✗				
10601950	✗	✗		✗	✗
10601949	✗	✗	✗	✗	
10602203	✗	✗		✗	✗

## **Peer Evaluation From For Group Work**

Evaluation Criteria	10602218	10602194	10601949	10601950	10602203
5 Attends group meetings regularly and arrives on time.	4	3	4	4	4
Contributes meaningfully to group discussions.	4	3	4	3	4
Completes the tasks on time.	4	4	4	4	3
Prepares work in a quality manner.	3	4	4	4	4
Contributes significantly to the success of the project in a cooperative and supportive attitude.	3	4	3	4	4
<b>TOTAL</b>	18	18	19	19	19

## **Plagiarism Report**





### PRIMARY SOURCES

---

- |   |   |     |
|---|---|-----|
| 1 | sqltechi.blogspot.com<br>Internet Source                              | 2%  |
| 2 | Submitted to Informatics Education Limited<br>Student Paper           | 1%  |
| 3 | Submitted to Colorado Technical University<br>Online<br>Student Paper | 1%  |
| 4 | Submitted to University of East London<br>Student Paper               | 1%  |
| 5 | Submitted to University of Plymouth<br>Student Paper                  | 1%  |
| 6 | getcompiled.com<br>Internet Source                                    | 1%  |
| 7 | o7planning.org<br>Internet Source                                     | <1% |
| 8 | www.hivmr.com<br>Internet Source                                      | <1% |
| 9 | Submitted to Gulf College Oman  |     |
-

10

Submitted to University of Lincoln

<1 %

Student Paper

11

Submitted to University of Greenwich

<1 %

Student Paper

12

Submitted to Kingston University

<1 %

Student Paper

Exclude quotes

On

Exclude matches

Off

Exclude bibliography

On