Coursework 1: Taxi company database development and tuning

Group members:

Names:	Student IDs:
Rayhannur Rohman:	170189255
Daniel Atewologun	180332788
Benjamin Arthur Bonful	180242650
Mohammed Mahfuzur Rahman	170401892

Contents:

Assumptions

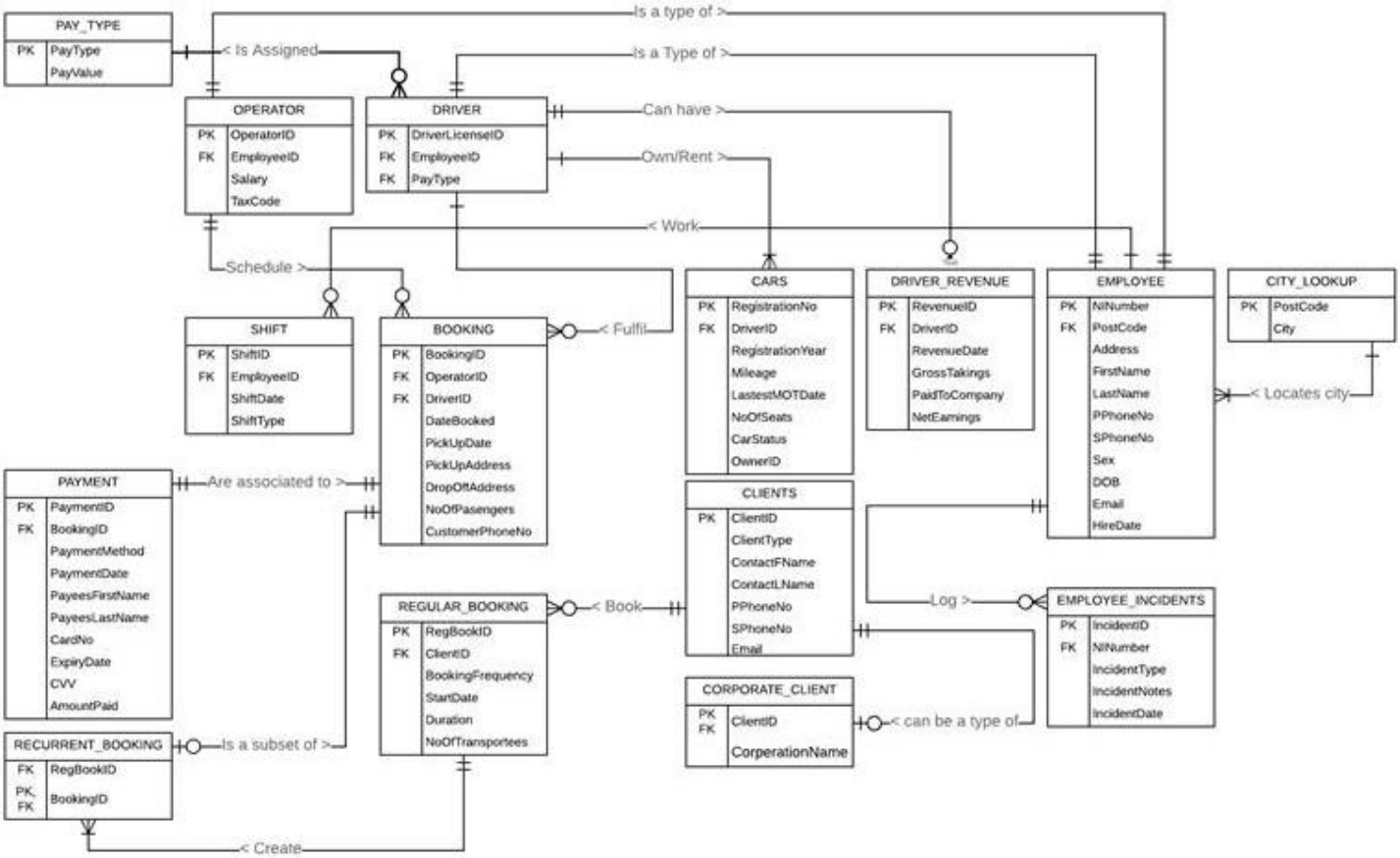
1. The taxi company operates exclusively in the UK.
    a. All employees must be able to work in the UK and must have a residence in the UK.
    b. All cars are assumed to be registered in the UK.
    c. All drivers have the certifications and licenses to work as a taxi driver in the UK.
2. All employees have to provide a phone number to their employers, so it can be used as their primary mode of contact.
3. Employees (drivers and operators) must have a primary phone number associated to them. Optionally, they can also provide a secondary phone number as an alternative mode of contact.
    a. An employee cannot be associated with a primary or secondary phone number that is already associated with another employee (Phone numbers must be unique).
4. Drivers must inform the company about the cars they own and will use for the business upon employment.
5. The system does not record data for staff that are not drivers or operators (such as cleaners and maintenance crews).
6. Employees and clients can only be associated with a single address.
    a. If they have more than one address, they must provide their primary address.
7. If an employee is terminated, resigns or retires, their information is removed from the database system.
    a. If they are rehired, they will be treated like a new employee - with a new recorded start date that reflects their rehire date and a flesh employee record
        i. Employees' records can be exported and stored separately for a period of time if they need to be reviewed in cases where former employees would like to be rehired after some time.
8. Sick days, vacation days and disciplinary actions are recorded in and denoted by an “Employee incidents” table.
    a. Employees are entitled to twenty-eight days of vacations.
    b. Sick days and vacation days must be at least a day long and the duration must be measured in days, not half days or hours (meaning a vacation cannot be booked for a few days and a few hours, they must be booked a certain amount of full days).
9. Drivers cannot be employed as operators and operators cannot be employed as drivers. Each individual employee is only hired to fulfill a single role and cannot be employed twice to fulfill the role of an operator and driver separately.
10. Drivers are considered to be privately contracted employees (self-employed), thus they file their taxes individually and are taxed based on their net earnings (which is the total amount of money they collect from customers in exchange for fulfilling bookings, minus the money they are obliged to pay the taxi company).
    a. In contrast, operators are considered salaried employees, so the system associates a tax code value to them - which the company's payroll department can use to file employee taxes.
    b. Employees are paid no more than “£1,000,000” a month.

11. The age of a car is derived from the car's registration number (its license place) which contains the year it was registered. This is in contrast to using a date that employees may claim to have purchased the car. An age derived from the purchase date of a car would not accurately reflect the car's actual age as it may have been used extensively before purchase.
    a. Each cars' mileage is recorded (and updated every year) to give a more accurate idea of its usage, which age cannot accurately indicate considering that a car can go unused and can still age.
12. Cars are either owned by the company (which rents cars to drivers) or by drivers.
    a. Drivers cannot drive cars they, or the company, do not own.
    b. Cars can only be rented to drivers who work for the company.
    c. Cars that are not associated with a driver or to the company are not recorded on the system.
13. The mileage count on a car (which can be found near or on a car's speed dials) is recorded and updated every month.
14. If a customer needs to transport more people than there are seats in the car with the highest number of seats, they would need to arrange multiple bookings
    a. If a regular booking requires the transportation of more people than there are seats in the car with the highest number of seats, multiple bookings will be created for them
15. Drivers can have more than a single car associated with them in the system.
    a. Drivers must have at least one car associated with them.
16. Drivers can either take a fixed, agreed upon, flat fee payment for each booking they fulfill, or they can take a percentage of each payment for each booking fulfillment, as their earnings.
17. Bookings should always have a driver and operator assigned to them on creation.
    a. The operator will be the operator to have registered the regular booking or the operator to have taken the call for the one-off customer booking (non-regular booking).
    b. If the operator or driver originally assigned to the booking is no longer recorded in the system (they no longer work for the company) the bookings can be in a state where they have no driver or operator
        i. All booking associated to driver must be fulfilled or reassigned before they are removed from the system.
18. Regular bookings can only be scheduled every month, week, every few days a week (e.g. Every weekend, every weekday or every select day(s) a week) or every day.
    a. After regular bookings are recorded in the system, bookings are created and assigned to drivers.
19. Shifts are scheduled from the first day of the next month to the last day and are recorded on the system on the day before the start of the month.
    a. Bookings for regular bookings are created on the system once shifts are created and assigned to drivers on the dates the regular booking encompasses.
20. The availability of employees is discussed before the creation of shifts to determine when they will work next month.
21. There are three shift types: Morning (6am - 2pm), Evening (2pm to 10pm) and Night (10pm - 6am)

22. The cost of the service offered via the booking must be calculated before the booking is fulfilled.
    a. The cost is calculated based on the distance from the pickup destination and the drop up destination and the estimated time it would take to travel between the two points with estimations of traffic taken into account.
23. Payments can be made with cash or card, but must be made between the creation of the a booking and its fulfillment
24. Bookings can only have a single payment associated to it
    a. Customers cannot pay partially through one mode of payments (e.g. cash) and partially through another mode of payment (e.g. debit card) - they must pay fully using one payment method.
25. Each booking must be paid for individually
    a. Bookings associated with regular bookings are paid for individually instead of being paid for collectively.
        i. Since bookings associated with regular bookings can be canceled, the individual payment for the booking can be refunded.
26. When bookings are canceled, the associated booking records are removed from the system.
27. If a client is a corporation, the name and contact details of the primary contact (the cooperate employee delegated to the job of requesting and managing bookings) is recorded.
    a. A private client is assumed to be a single person rather than a group of people.
    b. Private clients do not have a corporation name.
28. All employees are paid monthly.
29. The system does not record the history of payments made to operators (their monthly salary).
30. The system records the monthly earnings of drivers (payments they have collected through fulfilling bookings) and their costs (the amount they are obliged to contribute back to the taxi company to pay for operation costs and upkeep) as well as their net pay (their monthly earnings - their costs).
    a. Each “Revenue” record records the monthly earnings and takings of each driver.

Entity Relation diagram

Notation: Crow's foot
Normalisation: Third Normal Form
Keys: FK - Foreign Key, **PK** - Primary Key



Relational Schema

Keys:

- ***BoldItalicTableName*** (NonKeyAttribute, OtherNonKeyAttribute, ….)
    - **BoldAttribute** - Primary Key
    - UnderlinedAttribute - Foriegn Key

Employee (**NINumber**, PostCode, Address, FirstName, LastName, PPhoneNo, SPhoneNo Sex, DOB, Email, Hiredate)

Driver (**DriverLicenseID**, EmployeeID, PayType)

Operator (**OperatorID**, EmployeeID, Salary, TaxCode)

Employee_Incidents (**IncidentID**, NINumber, IncidentType, IncidentNotes, IncidentDate)

City_Lookup (**PostCode**, City)

Pay_Type (**PayType**, PayValue)

Shift (**ShiftID**, EmployeeID, ShiftDate, ShiftType)

Cars (**RegistrationNo**, DriverID, OwnedID, RegistrationDate, Mileage, LatestMOTDate, NoOfSeats, CarStatus)

Clients (**ClientID**, ContactFName, ContactLName, ClientType, PPhoneNo, SPhoneNo, Email)

Cooperate_Client (**ClientID**, CooperationName)

Booking (**BookingID**, OperatorID, DriverID, DateBooked, PickUpDate, PickUpAddress, DropOffAddress, NoOfPassengers, LuggageType, CustomerPhoneNo.)

Regular_Booking (**RegBookID**, ClientID, BookingFrequency, StartDate, Duration, NoOfPassengers)

Recurrent_Booking (RegBookID, **BookingID**)

Payment (**PaymentID**, BookingID, PaymentMethod, PaymentDate, PayeesName, CardNumber, ExpiryDate, CVV, AmountPaid)

Driver_Revenue (**RevenueID**, DriverID, RevenueDate, GrossTakings, NetEarnings, PaidToCompany)

Description and explanation of relations and relationships

EMPLOYEE, DRIVER and OPERATOR

The “EMPLOYEE” relation is a generalisation of the “DRIVER” and “OPERATOR ” relation - which respectively represent drivers and operators who work in the company. Its' attributes holds the information that both the relations share, including:

- Basic personal information:
    - Names (“Firstname” and “LastName”)
    - Date of births (“DOB”)
    - Genders (“SEX”)
- Contact details:
    - Primary and secondary phone numbers (“PPhoneNo” and “SPhoneNo”)
    - Email addresses (“Email”)
    - Residential addresses (“Address”, “PostCode” and “City”)
        - Where “Address” holds the first line of an address
- Basic information that employers usually collect about employees:
    - National insurance numbers (“NINumber”)
    - Hire dates (HireDate”)

“NINumber” attribute was selected to be the relation's primary key, as anyone in the UK, with the legal right to work, is guaranteed to have a unique government issued national insurance number. Alternatively, an attribute for a unique company issued “employee identifier” could have been created to identify each employee. However, the “NINumber” attribute already enables this function, so such an attribute would be redundant and would add unnecessary overhead to data creation transactions.

Since it is common for most people to possess a mobile phone as a primary mode of contact, and a landline as a secondary node of contact, a primary phone number attribute (“PPhoneNo”) and a secondary phone number attribute (“SPhoneNo”) exists within the relation to reflect this. Collecting primary and secondary phone numbers is common practice for employers - who use them as the primary method of contacting employees. Alternatively, a separate relation (with a one to many relationship with the “EMPLOYEE” relation) could have been created to store more than two phone numbers for each employee. Such an approach could eliminate the existence of null values in cases where employees do not have a secondary phone number. However, cases where employees would have more or less than 2 phone numbers would be rare, so such a decision would add unnecessary overhead when querying the system for an employee's contact details.

Drivers and Operators have an “EmployeeID” attribute in their respective specialised relations (“DRIVER” and “OPERATOR”) - which stores their national insurance number, the same value that their generalized employee record stores. It links the respective relations to the “EMPLOYEE” relation. However, it is not used as a primary key in either relation to avoid (or reduce the likelihood of) operators being associated with other relations that expect to have a driver associated with them (such as in the bookings relation which stores an operator's ID and a Driver's ID) and vice versa. The “DRIVER” relation stores each drivers' driver license number (“DriverLicenseID”) as its primary key attribute. This attribute will store a value that has a format that's distinctly different from the format used in the “OPERATOR”

relation's “OperatorID” primary key attribute. The difference is format will enable users and programs who interact with the system to easily identify mistakes in data entry with other relations where operators and drivers are concerned.

Since drivers and operators are employed compensated differently, operators have a “Salary” and “TaxCode” recorded in their relation (for the company's finance department to use to handle payroll) and drivers have a “PayType” attribute in their relation (which indicates how much they earn per booking fulfillment and can be used to calculate the company's costs and net profit in the “DRIVER_REVENUE” relation). Since drivers are contracted workers, there is no need to store a tax code value for them as they are expected to file taxes on their own.

The relationship between employees and both drivers and operators is a bilateral “one and only one” relationship, where drivers and operators records are a composition of a single employee record, and vice versa. Such a relationship implies that the deletion of an employee record should trigger the deletion of the driver or operator it represents and the deletion of a driver or operator should lead to the deletion of the employee record that represents it.

EMPLOYEE_INCIDENTS

The “EMPLOYEE_INCIDENTS” relation record employee sick days, the use of vacation days and disciplinary actions. The relations include:

- A unique primary key identifier for each record (“IncidentID”)
- The national insurance number that references the employee related to the record (“NINumber”)
- A type descriptor which denotes the type of incident that is recorded (“IncidentType”)
- An attribute where a detail description of the even can be stored (“IncidentNotes”)
- A date field which stores the date that the incident begun or, in the case of pre-booked vacations, the date it is set to begin (“IncidentDate”)

Employees can have zero or many incidents associated depending on their conduct, use of vacation and sick days and how new they are to the company. Each employee incident must be exclusively related to the employee it was originally created for (incidents cannot be reassigned to another employee). Therefore, the 2 relations have a composition “One and only one to zero or many” relationship.

PAY_TYPE

The “PAY_TYPE” relation denotes:

- The types of pay drivers can receive through its “PayType” primary key attribute.
- The rates for each pay type through its “PayValue” attribute

Drivers are assigned a “PayType” which indicates how they are paid per booking fulfillment. They can either be paid a fixed amount of money per booking fulfillment, or a percentage of the amount paid for each booking that they fulfilled. To improve the ease of changing the pay rate of each pay type and adding new pay types, the “PayType” relation was created to store the percentage or fixed amount of money that the taxi company has agreed to pay a set of drivers per booking fulfillment, through the “PayValue” attribute. This enables pay types to be extensible and allows pay types and rates to be updated more easily. The latter benefit is

possible with said approach since it will allow drivers to reference and share the same pay type and allows pay rates to be updated per pay type record instead of per driver. Since there are likely to be less paytypes than employees, this is more efficient, as less updates are required in a situation where a set of drivers need their pay rates changed.

While drivers must have a pay type assigned to them, there can exist pay types that are either not assigned to any driver or are assigned to multiple drivers, hence there is a aggregation "one and only one to zero or many" relationship between pay type and driver (where, notably, pay types are aggregations of drivers, not compositions - meaning paytypes can exist without drivers).

CITY_LOOKUP

The "CITY_LOOKUP" relation acts as a lookup table for cities that are associated to a given postcode. It includes:

- A unique primary key attribute for postcodes ("PostCode") which is used for the lookup process.
- A city field which stores the city associated to a post code ("City")

Prior to normalizing the initial design for the "EMPLOYEE" relation, the "CITY_LOOKUP" relation did not exist and its "City" field was in the "EMPLOYEE" relation. However, to get the employee relation to third normal form, the transitive relationship between the "Address" attribute, the "PostCode" attribute and the "City" attribute was eliminated through the creation of a new relation (the "CITY_LOOKUP" relation).

A set of postcodes and cities can be used by one or more employees, but each employee can only have a single post code and city associated to them - hence the aggregation "one and only one to one or many" relationship.

SHIFT

The "SHIFT" relation contains the shifts that are assigned to employees and information about each shift. It includes:

- A unique identifier for each shift ("ShiftID")
- A reference to the employee that is assigned to the shift ("EmployeeID")
- The date that the shift is sent to begin ("ShiftDate")
- An indication of when the shift begins and ends ("ShiftType")

Employees are set zero or many shifts to work per shift assignment, which are inserted into the shifts table at the end of every month. Each shift record records the employee assigned to it ("EmployeeID"), the date the shift will begin ("ShiftDate") and the shift type - which indicates if a shift is a morning shift (6am to 2pm), evening shift (2pm to 10pm) or night shift (10pm - 6am). Each shift has a unique identifier called "ShiftID" which enables the shift table to be in second normal form, as the table's non-prime attributes (all its other fields) will be fully dependent on this candidate key.

Employees can swap shifts, hence the "one and only one to zero many" aggregation relationship between shifts and employees.

CARS

The “CARS” relation holds information about the cars that drivers reportedly own and use to fulfill bookings, as well as cars that the company owns and rents to drivers. It includes:

- The car's registration number (“RegistrationNo”)
- The licence number of the driver who is using the car (“DriverID”)
- The year the car was register (“RegistrationYear”)
- The number miles that the car has traveled (“Mileage”)
- The date that the car last has an MOT check (“LatestMOTDate”)
- The number of seats the car has (“NoOfSeats”)
- The status of the car (“CarStatus”)
- The and Identifier for the car's owner (“OwnerID”)

Since all cars in the UK have a unique registration number, the cars relations has “RegistrationNo” as its primary key. While the registration number usually contains the last 2 digits of the year a car was registered, the decision to make it a separate attribute (“RegistrationDate”) was made to remove the need to process the “RegistrationNo” value to partially derive the registration year, and to allow the system to store the exact registration year instead of the last 2 digits of the registration year.
Since the date of registration does not always accurately reflect a car's usage (considering the fact that cars can be registered before being sold and can be used lightly or extensively), the “Mileage” attribute was added to the relation to record the amount of miles traveled by the car. This value is automatically recorded by all cars and displayed on the car's dashboard, so it is easy to attain.

For cars that are owned and rented by the company, the “OwnerID” will hold a string of characters that no existing driver's license will have (it will contain a special character that license does not have).

The relation has a foriegn key attribute, “DriverID”, which indicates which driver is using the car . Cars cannot exist (are not recorded by the system) if they are not associated with a driver that works for the company. This prevents the system from storing cars that do not relate to, or are no longer used by, the business. Thus the “DriverID” cannot store a null value.

CLIENTS and CORPORATE CLIENTS

The “CLIENTS” relation records information about the private and corporate clients that the company works with. Clients are assigned:

- A unique identifier primary key (“ClientID”)
- The primary contact's first and last name (“ContactFName” and “ContactLName”)
- A “type” to indicate if they are a private or a corporate client (“ClientType”)
- A primary and secondary phone number to contact the primary client through (“PPhoneNo” and “SPhoneNo”)
- An email address to use an alternative mode of contact (“Email”)

The information in the “CLIENTS” relation is sufficient for providing a representation of private clients, however corporate clients have a Corporation name (“CorporationName”) in this system. This would hold a redundant or empty value if it was stored in the clients table, as private clients do not have a corporation name. Therefore, a specialized relation for

corporate clients exist to ensure the “CorporationName” does not hold null values and to ensure all fields in the client relation are functionally dependant on the its “ClientID” primary key (to abide by the rules of second normal form) - since “CorporationName” would be functionally dependant on “ClientType” (a non-prime attribute) if the existed in the same table.

Considering the fact that not all clients are corporate clients, the participation in the relationship between the Clients relation and corporate client relation is optional. However, all corporate clients are considered to be clients, so there is a mandatory composition “one and only one to zero to one” relationship between the corporate clients relation and the client relation.

BOOKING

The “BOOKING” relation records information about each booking that is scheduled by an operator and assigned or fulfilled by a driver. For each booking the “BOOKING” relation records:

- A unique primary key identifier for each booking (“BookingID”)
- An reference to the operator who created the booking on a customer's behalf (“OperatorID”)
- A reference to the driver that is, or was, assigned to fulfill the booking (“DriverID”)
- The date the booking was created (“DateBooked”)
- The date and time the customer or client expects to be picked up (“PickUpDate”)
- The place the customer or client expects to be picked up from (“PickUpAddress”)
- The place the customer or client expects to be dropped off (“DropOffAddress”)
- The number of passengers that will or have been transported during the fulfillment of the booking (“NoOfPasengers”)
- The primary phone number of the customer or client that made the booking (“CustomerPhoneNo”)

The “PickUpAddress” and “DropOffAddress” of a booking record is intended to be entered into a road navigation application (such as google maps) that directs them to a destination from their current location. As such, there is no need to split the address into 3 lines of addresses (first line of address, postcode and city), since navigation applications take a single full address as input.

Customers, who are not clients, are not recorded on the system. Booking records simply take note of customers' phone numbers and that number is used to contact them and confirm their identity. Clients have their primary phone number stored in this field if the booking is created to fulfill a regular booking arranged by a client.

A single driver and a single operator is assigned to a booking and operators and drivers can be assigned to multiple bookings, hence the “one and only one to zero to many” relation between the two relations. While a booking can be assigned to a new driver, the operator who scheduled the booking will always be associated with the booking they created - hence the composition “one and only one” relationship between booking and operator. New drivers and operators will not have bookings associated with them, so the participation of their respective relations with the booking relation is optional.

RECURRENT_BOOKING and REGULAR_BOOKING

The “RECURRENT_BOOKING” relation stores each individual regular booking that a client requests and information about the regular booking - which can be used to determine the amount and types of bookings that need to be arranged to fulfill the request. For each regular booking that a customer schedule the relation records:

- A unique primary key identifier for each regular booking (“RegBookID”)
- A reference to the client that requested the booking (“ClientID”)
- An indication of how off the client would like a booking to be scheduled per a time interval (“BookingFrequency”)
- The date that the regular booking is scheduled to start (“StartDate”)
- The duration of the regular booking from the start date (“Duration”)
- The number of passengers that need to be transported during the fulfillment of each regular booking (“NoOfTransportees”)

Clients can request the company to make regular bookings for a duration of time. The regularity (the frequency of bookings per interval of time) can be specified and the date in which the customer wants the regular bookings to begin from can also be specified.
With the information in the “NoOfTransportees” attribute, an appropriate number of bookings can be scheduled (booking records can be created) to reflect the client's specific requirements. The same clients can schedule multiple regular bookings, each with different frequencies, start dates, durations and passenger count, hence the “one and only one to zero or many" relationship.

Since each booking is fulfilled by a single driver, each driver can only drive one car at a time and each car has a limited number of seats, the regular booking relation's “NoOfPassenger” is required to determine the number of bookings that need to be fulfill to transport the specified number of passengers. If a client would like 40 people to be transported and, at most, each car registered in the system has 5 seats (4 of which are free when the driver is accounted for), 10 or more bookings would need to be arranged to accommodate that number. So if the duration of a regular booking is a month, the frequency is twice a week, and each booking requires 40 people to be transport, 80 (20 x 4 x (40 / 4) - 2 times a week multiplied by the 4 week in a month multiplied by (40 passengers divided by 4 seats per booking)) or more bookings would be required to fulfill the booking - assuming there are at most 4 seats free in each available driver's car.

The “RECURRENT_BOOKING” relation was created to map a set of bookings records to an associated regular booking. While the “REGULAR_BOOKING” relation's primary key could have been placed in the booking relation (as a foreign key) to map such associations, creating a separate relation eliminates the potential for there to be null values. Booking records, that have not been created for the fulfillment of a regular booking, will have a null or redundant value in a hypothetical “RegBookID” foriegn key, since there would be no regular booking to identify. Therefore, a relation that records the “RegBookID” of each regular booking and the unique “BookingID” of every booking created for a regular booking is a better alternative when pursuing a normalized NF3 database design.

PAYMENT

The “PAYMENT” relation records payments that are collected for each scheduled or fulfilled booking. It records:

- A unique primary key attribute identifier for each payment (“PaymentID”)
- A reference to the booking that it is directly and exclusively linked to it (“BookingID”)
- The payment method used to make the payment (“PaymentMethod”) - which indicates the type of payment (cash payments, debit card payments, credit card payments, etc) that was received for a booking.
- An attribute that records the date that the payment was received (“PaymentDate”)
- The name of the individual who made the payment, or, if the payment was made by card, the name of the card holder (“PayeesName”)
- The 16-digit number of the card used to make the payment (“CardNumber”)
- The expiry date of the debit or credit card use for the payment (“ExpiryDate”)
- The 3-digit security code of the card used for payment (“CVV”)
- The amount of money that was expected and paid for the associated booking's fulfillment (“AmountPaid”)

Since the majority of payments are likely be made through the use of a debit or credit cards, the “CardNumber”, “ExpiryDate” and “CVV” fields have not been recorded in a separate relation - despite the fact that they could contain null values if the payment type is not credit or debit card. Creating such a relation would require the system to store credit card information, which is highly sensitive data that is likely undesirable for a database system that focuses on recording bookings, employees and company revenue.

Every booking is exclusively linked to a single payment record and every payment is exclusively linked to a single booking record - hence the composition “One and only to one and only one” relationship relationship between the 2 relations.

DRIVER_REVENUE

The “DRIVER_REVENUE” relation records the monthly takings (the raw amount of money that has been earned through the fulfillment of fulfilled booking) of each driver, along with the the costs that are deducted from that number (the amount that is contributed to the company) before the driver's monthly net income is calculated. The relation records:

- A unique primary key identifier for each record (“RevenueID”)
- A reference to the driver that the record references (“DriverID”)
- The date that the revenue report was generated (“RevenueDate”)
- The total amount of money earned from fulfilling bookings in the time span of a month (“GrossTakings”)
- The amount of money that was paid to the company for upkeep, maintenance, operators salaries etc. (“PaidToCompany”)
    - This includes the company's share of booking fulfillments - where drivers receive a sum of money for each booking fulfillment depending on their pay type and the company receives the remainder of the money.
- The amount of money that the driver has earned that month minus all the costs that were deducted from their gross takings (“NetEarnings”)

The company's monthly revenue (assuming that booking fulfillment is the only source of its revenue) can be derived from looking at the set of “DRIVER_REVENUE” records with a

“RevenueDate” that contains the month and year in question and finding the sum off the “GrossTakings” attribute. This is because each record contains the amount of money that each driver has brought in, so the revenue of all drivers in a given month is the revenue for that month. The data can be aggregated further to find yearly revenue, to work out company profits and cost and to determine the amount of money that is spent on paying drivers for their work.

Considering the fact that new drivers will not have any revenue to report and that revenue reports (“DRIVER_REVENUE” records) can only be generated and exclusively linked to a single driver, it makes sense to make the relationship between it and the “DRIVER” relation a composition “One and only one to zero to many” relationship.

SQL STATEMENTS

Statement 1

Description:
show the NETEARNINGS (Gross takings minus cost and contributions toward the company) of all drivers who earn more than £2500 in descending order.

Statement:

```
SELECT EMPLOYEE.NINUMBER, EMPLOYEE.FIRSTNAME, EMPLOYEE.LASTNAME,
DRIVER_REVENUE.NETEARNINGS
FROM employee
JOIN driver ON employee.NINUMBER=driver.EMPLOYEEID
JOIN DRIVER_REVENUE ON driver.driverLicenseID = DRIVER_REVENUE.DRIVERID
GROUP BY EMPLOYEE.NINUMBER, EMPLOYEE.FIRSTNAME, EMPLOYEE.LASTNAME,
NETEARNINGS
HAVING NETEARNINGS > 2500
ORDER BY NETEARNINGS DESC;
```

Screenshot:

NINUMBER	FIRSTNAME	LASTNAME	NETEARNINGS
EP123456Q	DAHLIA	EASTIN	2694
EP123457I	GARRY	HANSON	2692
EP123457B	FIDEL	GARFIELD	2683
EP123456L	CALISSA	DESMOND	2680
EP123457L	GARON	IRVING	2679
EP123457E	FERLIN	GORAN	2605
EP123456X	EDGAR	FITZ	2595
EP123457H	GALILEO	HANSON	2580
EP123457O	GARON	IKEDA	2566
EP123457G	GILDA	HAMILTON	2525
EP123456I	BAHULA	CADORETTE	2524
EP123456K	CADEN	DERWIN	2503

Download CSV

12 rows selected.

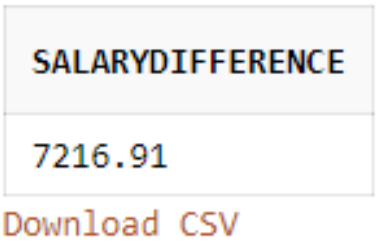
Statement 2

Description:
Shows the Difference between the average monthly income of drivers and operators.

Statement:

```
SELECT ROUND((AVG(OPERATOR.SALARY) - AVG(PAY_TYPE.PAYVALUE)),2) AS
SALARYDifference
FROM DRIVER
CROSS JOIN OPERATOR
JOIN DRIVER_REVENUE ON DRIVER_REVENUE.DRIVERID = DRIVER.DRIVERLICENSEID
JOIN PAY_TYPE ON PAY_TYPE.PAYTYPE = DRIVER.PAYTYPE
WHERE DRIVER.PAYTYPE IN (SELECT PAYTYPE
FROM PAY_TYPE
WHERE PAYTYPE IN ('FIXEDTEMP', 'FIXEDFULL'));
```

Screenshot:



SALARYDIFFERENCE
7216.91

Download CSV

Statement 3

Description:
Shows all employees name and IDs who have afternoon shifts in the week beginning 23rd of November.

Statement:

```
SELECT  Employee.FirstName, employee.LastName, SHIFTID, SHIFTTYPE,
SHIFTDATE
FROM employee
JOIN shift ON shift.employeeID=employee.NINUMBER
WHERE ShiftType = 'AFTERNOON' AND ShiftDate BETWEEN '23-NOV-2020' AND
'30-NOV-2020';
```

Screenshot:

FIRSTNAME	LASTNAME	SHIFTID	SHIFTTYPE	SHIFTDATE
BAHULA	CADORETTE	SH123456J	AFTERNOON	24-NOV-20
DALIYA	EASTON	SH123456R	AFTERNOON	28-NOV-20
EDWARD	FLOYD	SH123456B	AFTERNOON	25-NOV-20
FABIEN	GARNER	SH123456G	AFTERNOON	24-NOV-20

Download CSV
4 rows selected.

Statement 4

Description:
Shows all the names and phone numbers of all drivers who have 4 seater cars for bookings that take between 1 and 4 passengers.

Statement:

```
SELECT employee.FirstName, employee.LastName, employee.PPHONENO
FROM employee
JOIN driver ON employee.NINUMBER=driver.EMPLOYEEID
JOIN cars ON driver.driverLicenseID=cars.DRIVERID
JOIN booking ON booking.DRIVERID=driver.driverLicenseID
WHERE booking.NOOFPASSENGERS BETWEEN 1 AND 4 AND cars.NoOfSeats = 4;

/*Shows all drivers who have 2 and 3 seater cars with a postcode that
starts with an "E"*/
/*Selects the names and phone numbers of clients who have regular
bookings"*/
```

Screenshot:

FIRSTNAME	LASTNAME	PPHONENO
BAHULA	CADORETTE	07000000010
BAHULA	CADORETTE	07000000010
BARBRA	CADOGEN	07000000011
CAILYN	DIXIE	07000000017
DAGNA	EARWOOD	07000000018
DALLAS	EAKES	07000000021
ED	FAHL	07000000023
EDDY	FLETCHER	07000000025
EDDY	FLETCHER	07000000025
EDGAR	FITZ	07000000026
GALILEO	HANSON	07000000037
GALILEO	HANSON	07000000037
GARON	IIDA	07000000046

Download CSV
13 rows selected.

Statement 5

Description:
Presents drivers' employee information and displays how much they earn.

Statement:

```
SELECT DRIVER.EmployeeID, DRIVER.PayType, CARS.RegistrationYEAR,
CARS.NoOfSeats, CARS.CarStatus, PAY_TYPE.PayType, PAY_TYPE.PayValue
FROM PAY_TYPE
INNER JOIN DRIVER ON PAY_TYPE.PayType = DRIVER.PayType
INNER JOIN CARS ON CARS.DRIVERID = DRIVER.DRIVERLICENSEID
ORDER BY PAY_TYPE.PayValue ASC;
```

Screenshot:

EMPLOYEEID	PAYTYPE	REGISTRATIONYEAR	NOOFSEATS	CARSTATUS	PAYTYPE	PAYVALUE
EP123457O	FIXEDFULL	2012	6	AWAITING REPAIR	FIXEDFULL	36000
EP123457B	FIXEDFULL	2014	4	IN FOR SERVICE	FIXEDFULL	36000
EP123456L	FIXEDFULL	2011	4	IN FOR SERVICE	FIXEDFULL	36000
EP123457I	FIXEDFULL	2014	6	ROADWORTHY	FIXEDFULL	36000
EP123457H	FIXEDFULL	2012	4	IN FOR SERVICE	FIXEDFULL	36000
EP123456I	FIXEDFULL	2018	4	ROADWORTHY	FIXEDFULL	36000
EP123456Q	FIXEDFULL	2011	4	ROADWORTHY	FIXEDFULL	36000
EP123457E	FIXEDFULL	2015	4	ROADWORTHY	FIXEDFULL	36000
EP123456K	FIXEDFULL	2012	5	ROADWORTHY	FIXEDFULL	36000
EP123456X	FIXEDFULL	2010	4	ROADWORTHY	FIXEDFULL	36000
EP123457L	FIXEDFULL	2006	6	ROADWORTHY	FIXEDFULL	36000
EP123457G	FIXEDFULL	2007	4	ROADWORTHY	FIXEDFULL	36000
EP123456U	FIXEDTEMP	2018	4	ROADWORTHY	FIXEDTEMP	24000
EP123456S	FIXEDTEMP	2017	4	ROADWORTHY	FIXEDTEMP	24000
EP123457K	FIXEDTEMP	2008	4	AWAITING REPAIR	FIXEDTEMP	24000
EP123457A	FIXEDTEMP	2013	5	ROADWORTHY	FIXEDTEMP	24000
EP123456Z	FIXEDTEMP	2013	5	IN FOR SERVICE	FIXEDTEMP	24000
EP123457C	COMM80	2017	4	IN FOR SERVICE	COMM80	80
EP123456N	COMM80	2013	4	ROADWORTHY	COMM80	80
EP123457J	COMM75	2017	4	ROADWORTHY	COMM75	75
EP123457F	COMM75	2012	4	IN FOR SERVICE	COMM75	75
EP123457D	COMM75	2009	4	IN FOR SERVICE	COMM75	75
EP123456W	COMM75	2009	4	ROADWORTHY	COMM75	75
EP123456V	COMM75	2017	4	AWAITING REPAIR	COMM75	75
EP123456T	COMM75	2013	4	ROADWORTHY	COMM75	75
EP123456R	COMM75	2010	4	ROADWORTHY	COMM75	75
EP123456P	COMM75	2014	4	ROADWORTHY	COMM75	75
EP123456O	COMM75	2010	4	IN FOR SERVICE	COMM75	75
EP123456M	COMM75	2016	7	ROADWORTHY	COMM75	75
EP123456J	COMM75	2008	4	AWAITING REPAIR	COMM75	75
EP123457N	COMM75	2013	4	ROADWORTHY	COMM75	75
EP123457M	COMM70	2012	4	ROADWORTHY	COMM70	70

Download CSV
32 rows selected.

Statement 6

Description:
Displays a client's regular booking details such as booking frequency, duration and start date. Also displays which client has the most booking frequency

Statement:

```
SELECT REGULAR_BOOKING.RegBookID, REGULAR_BOOKING.ClientID,
MAX(REGULAR_BOOKING.BookingFrequency) AS BOOKINGFREQUENCY,
REGULAR_BOOKING.StartDate,REGULAR_BOOKING.Duration, CLIENTS.ClientID,
CLIENTS.ClientType,CLIENTS.PPhoneNo, CLIENTS.SPhoneNo, CLIENTS.Email
FROM REGULAR_BOOKING
INNER JOIN CLIENTS ON REGULAR_BOOKING.ClientID = CLIENTS.ClientID
GROUP BY REGULAR_BOOKING.RegBookID, REGULAR_BOOKING.ClientID,
REGULAR_BOOKING.StartDate,REGULAR_BOOKING.Duration, CLIENTS.ClientID,
CLIENTS.ClientType,CLIENTS.PPhoneNo, CLIENTS.SPhoneNo, CLIENTS.Email
ORDER BY CLIENTS.ClientType ASC;
```

Screenshot:

Employee Name	INCIDENTTYPE	INCIDENTNOTES	INCIDENTDATE	POSTCODE	CITY
GARRY HANSON	SICK	GOT MY FINGERS STUCK IN A BOWLING BALL	02-NOV-20	W10 4AD	WEST LONDON
CAITLYN DEXTER	VACATION REQUEST	I WANT TO GO ON MY HONEYMOON	10-NOV-20	NW8 9ZL	NORTH WEST LONDON
BALDWIN CADLE	DISCIPLINARY ACTION	CAME TO WORK DRUNK. LICENSE SUSPENDED UNTIL FURTHER NOTICE	02-OCT-20	N14 4AU	NORTH LONDON
EDGAR FITZ	SICK	MY GIRLFRIEND BIT ME IN A BAD PLACE	12-OCT-20	SW17 6AF	SOUTH WEST LONDON
GARRY HANSON	SICK	I HAVE A BLOCKED NOSE	15-MAY-20	W10 4AD	WEST LONDON
BAKI CADIZ	VACATION REQUEST	I HEAR SPAIN IS A GREAT PLACE TO FIND CARS, MIND IF I TAKE A LOOK MYSELF?	07-MAR-20	N14 4AU	NORTH LONDON
EDWARD FLOYD	SICK	DRANK TOILET WATER, I DONT FEEL SO GOOD	03-NOV-20	SW17 6AF	SOUTH WEST LONDON

Download CSV
7 rows selected.

Statement 7

Description:
Find all employee incidents only in London by quick postcode search:
This is a query to find all the employee incidents that have happened only in london. It works by not having to search all the postcodes beginning letters from A-Z and instead just searches through 4 beginning postcode letters found in London.

Statement:

```
SELECT EMPLOYEE_INCIDENTS.NINumber, EMPLOYEE_INCIDENTS.IncidentType,
EMPLOYEE_INCIDENTS.IncidentNotes, EMPLOYEE_INCIDENTS.IncidentDate,
EMPLOYEE.NINumber, CITY_LOOKUP.PostCode, CITY_LOOKUP.City
FROM EMPLOYEE_INCIDENTS
INNER JOIN EMPLOYEE ON EMPLOYEE_INCIDENTS.NINumber = EMPLOYEE.NINumber
INNER JOIN CITY_LOOKUP ON CITY_LOOKUP.POSTCODE = EMPLOYEE.POSTCODE
WHERE CITY_LOOKUP.PostCode LIKE 'B%' OR CITY_LOOKUP.PostCode LIKE 'N%'
OR
CITY_LOOKUP.PostCode LIKE 'NW%' OR CITY_LOOKUP.PostCode LIKE 'M%' OR
CITY_LOOKUP.PostCode LIKE 'SW%' OR CITY_LOOKUP.PostCode LIKE 'W%' OR
CITY_LOOKUP.PostCode LIKE 'EN%'
OR CITY_LOOKUP.PostCode LIKE 'L%';
```

Screenshot:

NINUMBER	FIRSTNAME	LASTNAME	ADDRESS	GROSSTAKINGS	PAIDTOCOMPANY	NETEARNINGS
EP123456A	AADI	BACCI	5 ALLINGTON ROAD	3333.33	0	3333.33
EP123456B	AALAM	BABEL	44 ALLINGTON ROAD	3541.67	0	3541.67
EP123456C	ABBA	BACCARI	39 ALLINGTON ROAD	3250	0	3250
EP123456D	ABDULLAH	BACCHUS	60 ALLINGTON ROAD	3000	0	3000
EP123456E	AARON	BACHMAN	90 ALLINGTON ROAD	3416.67	0	3416.67
EP123456F	BAKI	CADIZ	63 KEYSE ROAD	3416.67	0	3416.67
EP123456G	BALDWIN	CADLE	23 KEYSE ROAD	3333.33	0	3333.33
EP123456H	BABETTE	CADOTTE	17 KEYSE ROAD	3166.67	0	3166.67
EP123456I	BAHULA	CADORETTE	72 KEYSE ROAD	3000	476	2524
EP123456J	BARBRA	CADOGEN	88 KEYSE ROAD	2932	833	2099
EP123456K	CADEN	DERWIN	57 DUNSTAN ROAD	3000	497	2503
EP123456L	CALISSA	DESMOND	75 DUNSTAN ROAD	3000	320	2680
EP123456M	CAITLYN	DEXTER	12 DUNSTAN ROAD	3265	916.25	2348.75
EP123456N	CAI	DILLON	93 DUNSTAN ROAD	3004	700.8	2303.2
EP123456O	CAILYN	DIXIE	73 DUNSTAN ROAD	2761	690.25	2070.75
EP123456P	DAGNA	EARWOOD	17 ADELAIDE GROVE	2637	759.25	1877.75
EP123456Q	DAHLIA	EASTIN	12 ADELAIDE GROVE	3000	306	2694
EP123456R	DALILA	EAKES	14 ADELAIDE GROVE	3222	905.5	2316.5
EP123456S	DALLAS	EAKES	14 ADELAIDE GROVE	2000	467	1533
EP123456T	DALIYA	EASTON	19 ADELAIDE GROVE	2765	791.25	1973.75
EP123456U	ED	FAHL	32 ABBEY MEWS	2000	468	1532
EP123456V	EDD	FADEL	48 ABBEY MEWS	2675	768.75	1906.25
EP123456W	EDDY	FLETCHER	2 ABBEY MEWS	2572	743	1829
EP123456X	EDGAR	FITZ	50 ABBEY MEWS	3000	405	2595
EP123456Z	EDWARD	FLOYD	61 ABBEY MEWS	2000	342	1658
EP123457A	FABIEN	GARNER	43 OCEANA CLOSE	2000	470	1530
EP123457B	FIDEL	GARFIELD	93 OCEANA CLOSE	3000	317	2683
EP123457C	FARRAH	GARRICK	90 OCEANA CLOSE	3035	707	2328
EP123457D	FAYIZ	GEORGE	95 OCEANA CLOSE	2861	672.2	2188.8
EP123457E	FERLIN	GORAN	25 OCEANA CLOSE	3000	395	2605
EP123457F	GABBY	HALSEY	6 PARRY STREET	3303	925.75	2377.25
EP123457G	GILDA	HAMILTON	1 PARRY STREET	3000	475	2525
EP123457H	GALILEO	HANSON	46 PARRY STREET	3000	420	2580
EP123457I	GARRY	HANSON	81 PARRY STREET	3000	308	2692
EP123457J	GARON	HAYES	28 PARRY STREET	3319	929.75	2389.25
EP123457K	GARON	INGRAM	16 BRISTOW ROAD	2000	288	1712
EP123457L	GARON	IRVING	96 BRISTOW ROAD	3000	321	2679
EP123457M	GARON	IBACH	58 BRISTOW ROAD	2506	851.8	1654.2
EP123457N	GARON	IIDA	86 BRISTOW ROAD	2623	755.75	1867.25
EP123457O	GARON	IKEDA	48 BRISTOW ROAD	3000	434	2566

Download CSV
40 rows selected.

Statement 8

Description:
Displays an invoice statement for each client, detailing their name, payment method, card number and the amount paid.

Statement:

```
SELECT CLIENTS.CONTACTFNAME, CLIENTS.CONTACTLNAME,
PAYMENT.PAYMENTMETHOD, PAYMENT.CARDNUMBER, PAYMENT.AMOUNTPAID FROM
CLIENTS
JOIN REGULAR_BOOKING ON CLIENTS.CLIENTID = REGULAR_BOOKING.CLIENTID
JOIN RECURRENT_BOOKING ON REGULAR_BOOKING.REGBOOKID =
RECURRENT_BOOKING.REGBOOKID
JOIN BOOKING ON RECURRENT_BOOKING.BOOKINGID = BOOKING.BOOKINGID
JOIN PAYMENT ON BOOKING.BOOKINGID = PAYMENT.BOOKINGID;
```

Screenshot:

CONTACTFNAME	CONTACTLNAME	PAYMENTMETHOD	CARDNUMBER	AMOUNTPAID
EDWARD	ELRIC	CARD	5420754499865718	7.65
RINTAROU	OKABE	CARD	5275496342325815	5.42
L.	LAWLIET	CARD	5136226533936057	4.17
ITACHI	UCHIHA	CARD	4415655174552366	51.66
HACHIMAN	HIKIGAYA	CARD	5213211675421049	37.9
TODOROKI	SHOTO	CARD	4671524735151580	6.56
IZUKU	MIDORIYA	CARD	4524075348975567	38.29
LEVI	ACKERMAN	CARD	4985301649478610	5.04
KURISU	MAKISE	CARD	5127071833675349	39.3
KAZUTO	KIRIGAYA	CARD	4432491216876558	14.43
KAZUTO	KIRIGAYA	CASH	-	15.46
GINTOKI	SAKATA	CARD	5177185476318618	4.54
LELOUCHE	LAMPEROUGE	CARD	5445471841716637	39.84
ZORO	RORONOA	CARD	4966118739314742	14.06
KILLUA	ZOLDYCK	CARD	5416855454432331	29.08

Download CSV
15 rows selected.

Statement 9

Description:
Displays a list of all clients who have an active regular booking.

Statement:

```
SELECT CLIENTS.CONTACTFNAME, CLIENTS.CONTACTLNAME FROM CLIENTS
WHERE EXISTS ( SELECT 1 FROM REGULAR_BOOKING
WHERE REGULAR_BOOKING.CLIENTID = CLIENTS.CLIENTID);
```

Screenshot:

CONTACTFNAME	CONTACTLNAME
EDWARD	ELRIC
RINTAROU	OKABE
L.	LAWLIET
ITACHI	UCHIHA
HACHIMAN	HIKIGAYA
TODOROKI	SHOTO
IZUKU	MIDORIYA
LEVI	ACKERMAN
KURISU	MAKISE
KAZUTO	KIRIGAYA
GINTOKI	SAKATA
LELOUCHE	LAMPEROUGE
ZORO	RORONOA
KILLUA	ZOLDYCK

Download CSV
14 rows selected.

Statement 10

Description:
Displays the name and national insurance number (NINUMBER) of the driver with the highest gross earnings.

Statement:

```
SELECT EMPLOYEE.NINUMBER, EMPLOYEE.FIRSTNAME, EMPLOYEE.LASTNAME,
DRIVER_REVENUE.GROSSTAKINGS FROM EMPLOYEE
JOIN DRIVER ON EMPLOYEE.NINUMBER = DRIVER.EMPLOYEEID
JOIN DRIVER_REVENUE ON DRIVER.DRIVERLICENSEID = DRIVER_REVENUE.DRIVERID
WHERE DRIVER.DRIVERLICENSEID =
(SELECT DRIVER_REVENUE.DRIVERID FROM DRIVER_REVENUE
GROUP BY DRIVER_REVENUE.DRIVERID HAVING
SUM(DRIVER_REVENUE.GROSSTAKINGS) =
(SELECT MAX(SUM(DRIVER_REVENUE.GROSSTAKINGS)) FROM DRIVER_REVENUE GROUP
BY DRIVER_REVENUE.DRIVERID));
```

Screenshot:

NINUMBER	FIRSTNAME	LASTNAME	GROSSTAKINGS
EP123457J	GARON	HAYES	3319

Download CSV

Statement 11

Description:
Selects the number of drivers who have a booking in August.

Statement

```
SELECT COUNT(DISTINCT Driver.DriverLicenseID) AS NUMOFDRIVERS FROM
BOOKING
INNER JOIN DRIVER ON DRIVER.DRIVERLICENSEID = BOOKING.DRIVERID
WHERE PICKUPDATE BETWEEN '01-AUG-2020' AND '31-AUG-2020';
```

Screenshot:

NUMOFDRIVERS
3

Download CSV

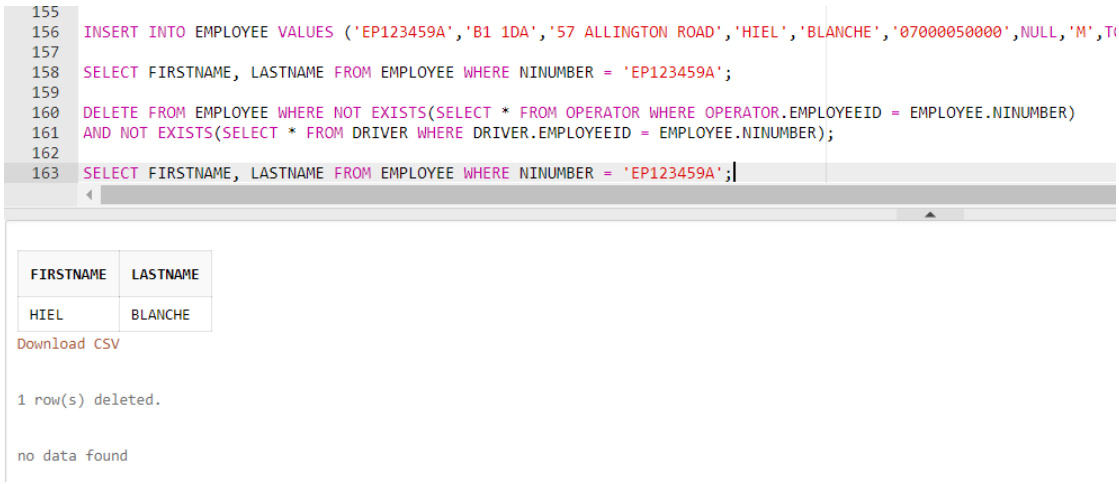
Statement 12

Description:
Deletes all employee records which do not have a driver or operator associated to them, a useful query to reduce redundancy within the system. In the current system the query shows there is no such redundancy.

Statement:

```
DELETE FROM EMPLOYEE WHERE NOT EXISTS(SELECT * FROM OPERATOR WHERE
OPERATOR.EMPLOYEEID = EMPLOYEE.NINUMBER)
AND NOT EXISTS(SELECT * FROM DRIVER WHERE DRIVER.EMPLOYEEID =
EMPLOYEE.NINUMBER);
```

Screenshot:



Triggers

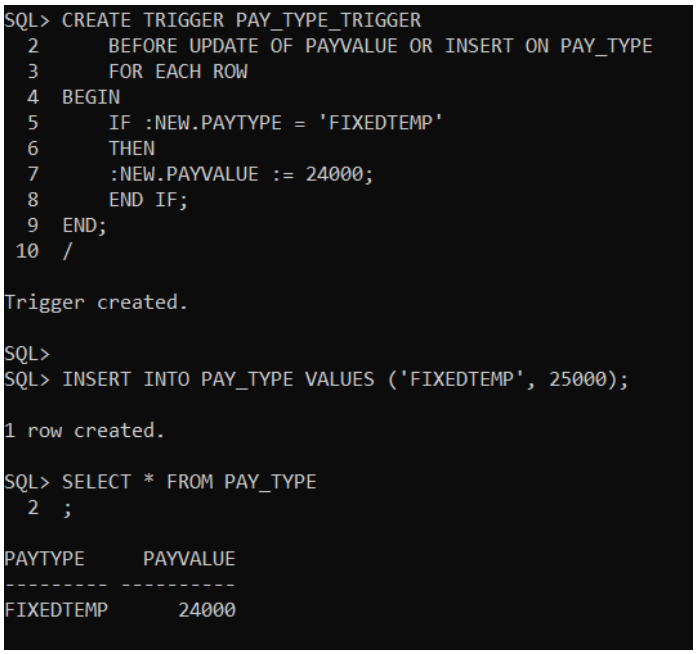
Trigger 1

Description:
Triggers when a new 'PayType' of type 'fixedtemp' is added to the pay_type table and the value of the 'fixedtemp' is not already 24000; the value is changed to 24000.

Trigger:

```
CREATE TRIGGER PAY_TYPE_TRIGGER
BEFORE UPDATE OF PAYVALUE OR INSERT ON PAY_TYPE
FOR EACH ROW
BEGIN
        IF :NEW.PAYTYPE = 'FIXEDTEMP'
        THEN
            :NEW.PAYVALUE := 24000;
        END IF;
END;
/
```

Screenshot:



```
SQL> CREATE TRIGGER PAY_TYPE_TRIGGER
  2      BEFORE UPDATE OF PAYVALUE OR INSERT ON PAY_TYPE
  3      FOR EACH ROW
  4  BEGIN
  5      IF :NEW.PAYTYPE = 'FIXEDTEMP'
  6      THEN
  7      :NEW.PAYVALUE := 24000;
  8      END IF;
  9  END;
 10  /

Trigger created.

SQL>
SQL> INSERT INTO PAY_TYPE VALUES ('FIXEDTEMP', 25000);

1 row created.

SQL> SELECT * FROM PAY_TYPE
  2  ;

PAYTYPE     PAYVALUE
--------- ----------
FIXEDTEMP      24000
```

Trigger 2

Description:
Trigger that fires when the date of the last MOT is sometime in the future.

Trigger:

```
CREATE TRIGGER CHECK_MOT_TRIGGER
    BEFORE UPDATE OF LASTMOTDATE OR INSERT ON CARS
    FOR EACH ROW
BEGIN
    IF :NEW.LASTMOTDATE > SYSDATE
    THEN
        RAISE_APPLICATION_ERROR(-20512, 'MOT DATE MUST BE LESS THAN
            TODAYS DATE');
    END IF;
END;
/
```

Screenshot:

```
no rows selected

SQL> CREATE TRIGGER CHECK_MOT_TRIGGER
  2  BEFORE UPDATE OF LASTMOTDATE OR INSERT ON CARS
  3  FOR EACH ROW
  4  BEGIN
  5  IF :NEW.LASTMOTDATE > SYSDATE
  6  THEN
  7      RAISE_APPLICATION_ERROR(-20512, 'MOT DATE MUST BE LESS THAN TODAYS DATE');
  8  END IF;
  9  END;
 10  /

Trigger created.

SQL> INSERT INTO CARS VALUES('CR1235G','DV123457G',2012,130650,TO_DATE('16-NOV-2020','DD-MON-YYYY'),6,'AWAITING REPAIR','DV123457G');
INSERT INTO CARS VALUES('CR1235G','DV123457G',2012,130650,TO_DATE('16-NOV-2020','DD-MON-YYYY'),6,'AWAITING REPAIR','DV123457G')
            *
ERROR at line 1:
ORA-20512: MOT DATE MUST BE LESS THAN TODAYS DATE
ORA-06512: at "DBCW1.CHECK_MOT_TRIGGER", line 4
ORA-04088: error during execution of trigger 'DBCW1.CHECK_MOT_TRIGGER'
```

Trigger 3

Description:
A trigger to ensure that the date of the last MOT is within the last 12 months. In the event it's not, the status of the car being inserted is changed. If it's been more than 4 years since the last MOT, then the insert is rejected.

Trigger:

```
CREATE OR REPLACE TRIGGER MOT_TRIGGER
BEFORE UPDATE OF LASTMOTDATE OR INSERT ON CARS
FOR EACH ROW
DECLARE MOT_AGE NUMBER;
BEGIN
SELECT (MONTHS_BETWEEN(SYSDATE, :NEW.LASTMOTDATE)/12)
INTO MOT_AGE FROM DUAL;
        IF (MOT_AGE > 1 AND MOT_AGE < 4)
        THEN
            :NEW.CARSTATUS := 'AWAITING REPAIR (MOT)';
            dbms_output.put_line('MOT is outdated');
        ELSE IF MOT_AGE > 5 THEN
            RAISE_APPLICATION_ERROR(-20532, 'MOT is outdated');
    END IF;
    END IF;
END;
/
```

Screenshots:

```
SQL> CREATE OR REPLACE TRIGGER MOT_TRIGGER
  2  BEFORE UPDATE OF LASTMOTDATE OR INSERT ON CARS
  3  FOR EACH ROW
  4  DECLARE MOT_AGE NUMBER;
  5  BEGIN
  6      SELECT (MONTHS_BETWEEN(SYSDATE, :NEW.LASTMOTDATE)/12)  INTO MOT_AGE FROM DUAL;
  7      IF (MOT_AGE > 1 AND MOT_AGE < 4)
  8      THEN
  9      :NEW.CARSTATUS := 'AWAITING REPAIR (MOT)';
 10      dbms_output.put_line('MOT is outdated');
 11      ELSE IF MOT_AGE > 5 THEN
 12      RAISE_APPLICATION_ERROR(-20532, 'MOT is outdated');
 13    END IF;
 14    END IF;
 15  END;
 16  /

Trigger created.
```

```
SQL> CREATE OR REPLACE TRIGGER MOT_TRIGGER
  2  BEFORE UPDATE OF LASTMOTDATE OR INSERT ON CARS
  3  FOR EACH ROW
  4  DECLARE MOT_AGE NUMBER;
  5  BEGIN
  6      SELECT (MONTHS_BETWEEN(SYSDATE, :NEW.LASTMOTDATE)/12)  INTO MOT_AGE FROM DUAL;
  7      IF (MOT_AGE > 1 AND MOT_AGE < 4)
  8      THEN
  9      :NEW.CARSTATUS := 'AWAITING REPAIR (MOT)';
 10      dbms_output.put_line('MOT is outdated');
 11      ELSE IF MOT_AGE > 5 THEN
 12      RAISE_APPLICATION_ERROR(-20532, 'MOT is outdated');
 13    END IF;
 14    END IF;
 15  END;
 16  /

Trigger created.

SQL> INSERT INTO CARS VALUES('CR1238Z','DV123457E',2012,87452,TO_DATE('01-APR-2017','DD-MON-YYYY'),4,'ROADWORTHY','DV123457E');

1 row created.

SQL> SELECT CARSTATUS FROM CARS
  2  WHERE REGISTRATIONNO = 'CR1238Z';

CARSTATUS
------------------------------
AWAITING REPAIR (MOT)
```

Trigger 4

Description:
Prevents disciplinary action incidents from having no notes/description.

Trigger:

```
CREATE TRIGGER INCIDENT_NOTES_TRIGGER
BEFORE UPDATE OF INCIDENTNOTES OR INSERT ON EMPLOYEE_INCIDENTS
FOR EACH ROW
BEGIN
IF :NEW.INCIDENTTYPE = 'DISCIPLINARY ACTION' AND :NEW.INCIDENTNOTES
IS NULL;
        THEN
                RAISE_APPLICATION_ERROR (-20001,'Incident notes must be
filled out');
END IF;
END;
/
```

Screenshot:

```
SQL> CREATE TRIGGER INCIDENT_NOTES_TRIGGER
  2      BEFORE UPDATE OF INCIDENTNOTES OR INSERT ON EMPLOYEE_INCIDENTS
  3      FOR EACH ROW
  4  BEGIN
  5      IF :NEW.INCIDENTTYPE = 'DISCIPLINARY ACTION' AND :NEW.INCIDENTNOTES IS NULL
  6      THEN
  7          RAISE_APPLICATION_ERROR (-20001,'Incident notes must be filled out');
  8  END IF;
  9  END;
 10  /

Trigger created.

SQL> INSERT INTO EMPLOYEE_INCIDENTS VALUES ('IN123456C','EP123456G','DISCIPLINARY ACTION','',TO_DATE('02-OCT-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES ('IN123456C','EP123456G','DISCIPLINARY ACTION','',TO_DATE('02-OCT-2020','DD-MON-YYYY'))
            *
ERROR at line 1:
ORA-20001: Incident notes must be filled out
ORA-06512: at "DBCW1.INCIDENT_NOTES_TRIGGER", line 4
ORA-04088: error during execution of trigger 'DBCW1.INCIDENT_NOTES_TRIGGER'
```

Trigger 5

Description:
Trigger to ensure that all employee incidents are of the type 'sick', 'disciplinary action' or 'vacation request'.

Trigger:

```
CREATE TRIGGER INCIDENT_TYPE_TRIGGER
BEFORE UPDATE OF INCIDENTTYPE OR INSERT ON EMPLOYEE_INCIDENTS
FOR EACH ROW
BEGIN
IF :NEW.INCIDENTTYPE != 'SICK' OR :NEW.INCIDENTTYPE != 'DISCIPLINARY
ACTION' OR :NEW.INCIDENTTYPE != 'VACTION'
THEN
RAISE_APPLICATION_ERROR(-20592, 'Invalid incident type');
END IF;
END;
/
```

Screenshot:

```
SQL> CREATE TRIGGER INCIDENT_TYPE_TRIGGER
  2  BEFORE UPDATE OF INCIDENTTYPE OR INSERT ON EMPLOYEE_INCIDENTS
  3  FOR EACH ROW
  4  BEGIN
  5  IF :NEW.INCIDENTTYPE != 'SICK' OR :NEW.INCIDENTTYPE != 'DISCIPLINARY ACTION' OR :NEW.INCIDENTTYPE != 'VACTION'
  6  THEN
  7  RAISE_APPLICATION_ERROR(-20592, 'Invalid incident type');
  8  END IF;
  9  END;
 10  /

Trigger created.

SQL> INSERT INTO EMPLOYEE_INCIDENTS VALUES ('IN123456D','EP123456G','NOTHING','TESTING TESTING 123',TO_DATE('02-OCT-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES ('IN123456D','EP123456G','NOTHING','TESTING TESTING 123',TO_DATE('02-OCT-2020','DD-MON-YYYY'))
            *
ERROR at line 1:
ORA-20592: Invalid incident type
ORA-06512: at "DBCW1.INCIDENT_TYPE_TRIGGER", line 4
ORA-04088: error during execution of trigger 'DBCW1.INCIDENT_TYPE_TRIGGER'
```

Trigger 6

Description:
Limits the number of records allowed in the operator table to 8.

Trigger:

```
CREATE OR REPLACE TRIGGER eight_operators_only
BEFORE INSERT ON OPERATOR
DECLARE OperatorCount NUMBER;
BEGIN
SELECT count(*) INTO OperatorCount FROM OPERATOR;
IF (OperatorCount > 7) THEN
RAISE_APPLICATION_ERROR(-20512, 'There are already 8 operators in the
system, you cannot hire or record anymore.');
        END IF;
END;
/
```

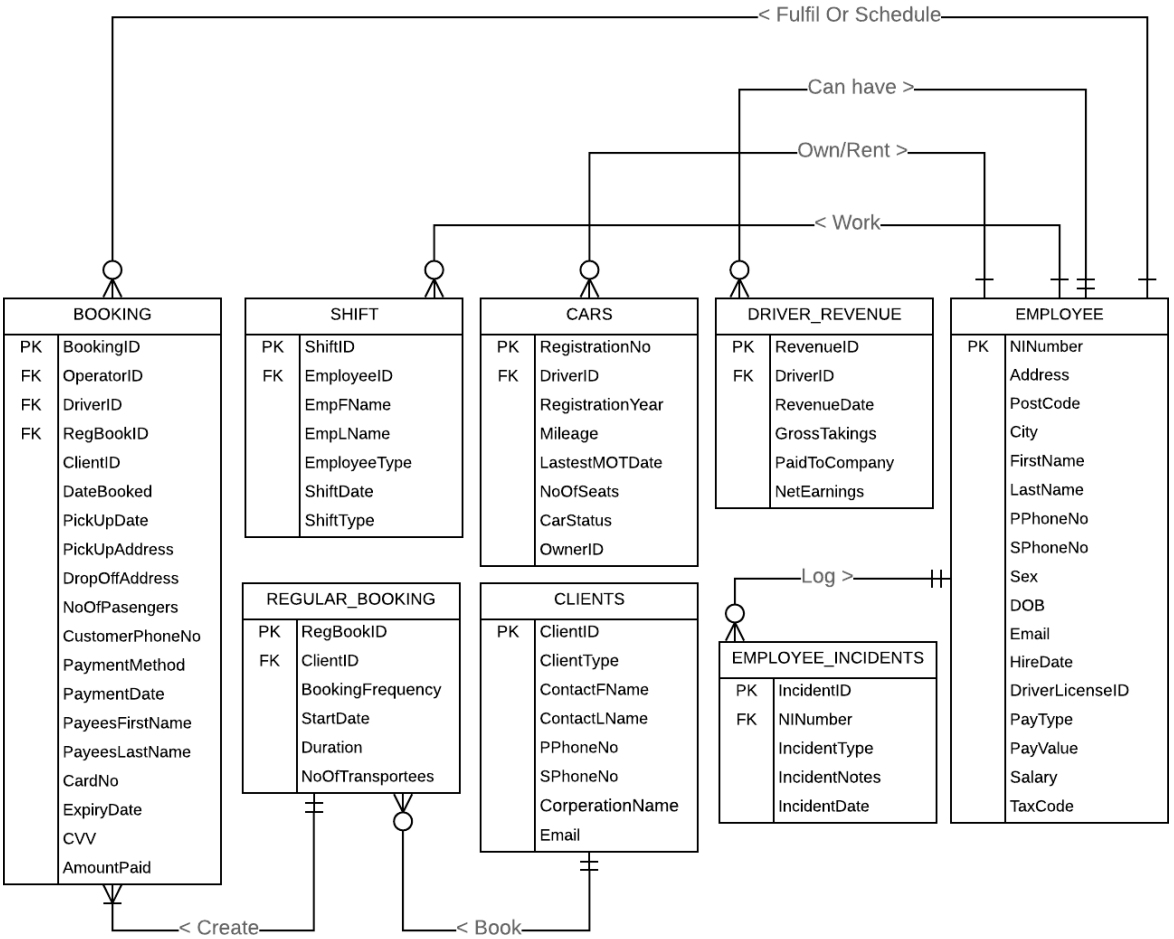
Screenshot:

```
SQL> CREATE OR REPLACE TRIGGER eight_operators_only
  2  BEFORE INSERT ON OPERATOR
  3  DECLARE OperatorCount NUMBER;
  4  BEGIN
  5      SELECT count(*) INTO OperatorCount FROM OPERATOR;
  6      IF (OperatorCount > 7) THEN
  7          RAISE_APPLICATION_ERROR(-20512, 'There are already 8 operators in the system, you can not hire or record anymore.');
  8      END IF;
  9  END;
 10  /

Trigger created.

SQL> INSERT INTO OPERATOR VALUES('OP123456I','EP123456H',38000,'1250L');
INSERT INTO OPERATOR VALUES('OP123456I','EP123456H',38000,'1250L')
            *
ERROR at line 1:
ORA-20512: There are already 8 operators in the system, you can not hire or
record anymore.
ORA-06512: at "DBCW1.EIGHT_OPERATORS_ONLY", line 5
ORA-04088: error during execution of trigger 'DBCW1.EIGHT_OPERATORS_ONLY'
```

Performance Tuning

Denormalization:

Denormalized conceptual design (ERD):



During the process of normalizing our initial and original conceptual designs for this database system, new entities were introduced to resolve many to many relationships and to separate attributes that could potentially hold null values in certain situations, into new or existing entities which had an optional relationship with the original entity. Attributes with transitive dependencies and sets of attributes that depended on non-prime attributes were also separated in the pursuit of creating a conceptual design that was in third normal form. This resulted in a design which has very few situations where redundant data would exist in the system. For example, in out design, “CLIENTS” object that represent private clients (as opposed to corporate client) do not need to hold a redundant value for the “Corporate name” attribute (which existed in the “CLIENTS” entity prior to normalization) because extra information about corporate clients are held in the separately created “CORPERATE_CLIENT” entity.

The denormalization process has also resulted in an implementable design where updates to certain shared information can be executed more efficiently, as it allows generalized information about specific entities to be stored in separate referenceable entity - and the information in such entities could be updated and made to reflect changes intended to affect a set of objects which reference the entity. An example of this is the “PAY_TYPE” entity,

which holds the pay rates (in the “PayValue” attribute), where pay rates are associated to assignable pay types and where pay types represent a pay rate that a defined set of drivers share. If all drivers with the same pay type are entitled to a raised pay rate, then the “PayValue” attribute of their assigned “PayType” can be updated to reflect that decision - as opposed to the alternative of updating a hypothetical “PayType” and “PayValue” attribute that exist in each instance of a driver entity - where the pay type value matches the pay type that needs to be updated. This means that transactions and statements that update pay types can be more efficient, as, in situations such as the aforementioned situations, they can update a single “Pay_Type” object instead of multiple “DRIVER” objects.

While highly normalized database systems can be updated more efficiently, querying data such systems can lack efficiency when compared to their original pre-normalized state. This is because the normalization process introduces many new entities, which increase the need for joins (which require a lot more extra processing - depending on the size of relations) to query associated, but separately stored, information about an entity. For example, the “DRIVER” and “PAY_TYPE” tables need to be joined to find each individual driver's pay rate in the normalized system. If the pay rate attribute was stored in the driver entity, then there would be no need for a join. Since the time complexity of transitions involving table joins can grow exponentially based on the amount of data on the table involved in the join and the number of tables that are being joined, reducing the need for them is desirable for systems that are expected to hold large datasets. The process of denormalization is used to reduce or eliminate the need for table joins and to find a good balance between maintaining a design that enables efficient table updates but also doesn't hinder the efficiency of commonly executed table queries.

We denormalized our original conceptual and relational design in the following manner:

- We combined relations that had “one to one” relationship and were likely to be joined for queries the taxi company is likely to need. The combined relations included:
- The “PAYMENT” relation, which was combined with the “BOOKING” relation.
    - Payments made on certain booking is something that is likely to be queried often and are unlikely to be updated often in such a system - so merging the tables would likely have a net positive benefit on the database system's performance.
- The “OPERATOR” and “DRIVER” relations, which was combined with the “EMPLOYEE” relation.
    - Drivers and operators do not have many difference in terms of the type of information that is recorded about them, so it makes sense to combine them into their generalization for the sake of improving performance - as doing so would reduce the need to join the specialized entities with their generalization (the “EMPLOYEE” relation) if further information about drivers and operators are required.
- The “CORPORATE_CLIENT” relation, which was combined with the “CLIENTS” relation.
    - Since corporate clients are distinguished from private clients solely by the fact that they have a corporation name, combining the “CORPORATE_CLIENT” entity with the client entity likely leads to a net improvement in general performance, as only a single potential redundant value is introduced to the client field in exchanged for the need for a join to view corporate client

information being eliminated. Viewing the full information of corporate clients is likely to be a common query considering the fact that the full information of clients is often needed to enable communication between companies and clients. Therefore, denormalizing the data in this instance is likely to be impactful.

- To further reduce the need for joins, we combined the “CITY_LOOKUP” relation with the “EMPLOYEE” relation
    - When looking for an employee's address, the city associated with their postcode is required in situations where the full address is required, thus combining the table with the employee table eliminates the need for joins in such a situation.
- We also combined the “PAY_TYPE” relation with the “DRIVER” relations, which was later combined with the “EMPLOYEE” relation.
    - Pay types will likely rarely be updated for a large group of drivers (there seem to be no logical reason to increase their rates in such a manner), so combining it with driver and then employee allows it to be queried more easily without a huge impact on update performance.
- We duplicated the foreign keys on the many sides of a one to many relationship to reduce the need for joins.
    - The “RECURRENT_BOOKING” relation was eliminated. The “RegBookID” attributed was put in the “BOOKING” relations and the
    - “REGULAR_BOOKING” relations' “ClientID” reference was placed in the booking relation to make queries that look for specific client's booking (by their ClientID) easier to create and less reliant on join statements. While this will drastically increase the number of bookings with null attribute values (since most bookings are expected to be one off booking made by non-clients and there will be no clientID value for such bookings), it will make the task of querying and selecting bookings that are associated with clients more efficient.
- The original design already introduced repeating groups of attributes like phone numbers, with the assumption that the people recorded in the system would likely only have two phone numbers. This design decision makes the original design already partially denormalized.
- The names of employees and an indication of the type of employee they are (Driver or Operator) was were added to the shift entity to make it easier to identify which employee is working a scheduled shift and to eliminate the need to join the employee and shift tables to know who is specifically meant to be working a shift. A query to see the name of an employee working a shift is likely to be a commonly executed query if the system is used to check which employees are on shifts.

The resulting conceptual and relational model for the denormalized version of the original third normal form database system has “BOOKING” and “EMPLOYEE” entities with many fields, which will likely make updates and create statements more inefficient as more data is required for each new record, and some of that data is likely to be redundant.

To test the denormalized design, we constructed both a normalized and denormalized versions of the database system to compare the result of running similar queries (which produce the same outcome) on both systems. The fields on the respective tables only had their primary key and field data types defined. Constraints were not added to the table as it would hinder the process of quickly creating large data sets to test the queries on. Since modern computers can execute complicated queries on small tables in a short amount of time, we needed to construct tables with enough data to get a recordable indication of the elapsed time between a query's execution and its completion. We also needed to make sure that there was not an unreasonable amount of data to query as the queries for the normalized system were going to use table joins, and table join can increase the number of data that is being processed by a transaction exponentially in certain situations.

We decided that if a query took more than 5 minutes to execute, we would cancel the transaction and attempt to run the query on a smaller, more reasonable dataset.

We also decided to create 3 to 6 insert statements (records) for each table and to then insert the table's contents within itself until the table grew to a certain length to create the proportion of data that we needed to get a reading of how long the execution of statements took on large tables.

Since the initial execution of a query can sometimes be slow due to an initial lack of cached data for the process, we decided to show the results of running the queries 5 times, from the first run to the fifth, and the average run time based on those 5 runs.

Denormalization vs Normalization Test 1:
Both queries essentially show the names, employee IDs and the total number of nights or afternoon worked by each employee that has worked night and afternoon shifts. It then orders the result based on the employeeID's alphabetical order.

Query for denormalized system:

```
/* DNORM QUERY 1 */
SELECT EmployeeID AS "Employee ID", EmpFName AS "First Name", EmpLName
AS "Last Name"
FROM SHIFT_DNORM_BIG
WHERE ShiftType IN ('NIGHT','AFTERNOON')
GROUP BY EmployeeID, EmpFName, EmpLName
ORDER BY EmployeeID;
```

Query for normalized system:

```
/* NORM QUERY 1 */
SELECT SHIFT_BIG.EmployeeID AS "Employee ID", EMPLOYEE_BIG.FirstName AS
"First Name", EMPLOYEE_BIG.LastName AS "Last Name"
FROM SHIFT_BIG
INNER JOIN EMPLOYEE_BIG ON SHIFT_BIG.EmployeeID = EMPLOYEE_BIG.NINumber
WHERE SHIFT_BIG.ShiftType IN ('NIGHT','AFTERNOON')
GROUP BY SHIFT_BIG.EmployeeID, EMPLOYEE_BIG.FirstName,
EMPLOYEE_BIG.LastName
ORDER BY SHIFT_BIG.EmployeeID;
```

	Run 1	Run 2	Run 3	Run 4	Run 5	Average
Normalized Table - 393216 Rows of data is table	Beyond 5 minutes	Beyond 5 minutes	Beyond 5 minutes	Beyond 5 minutes	Beyond 5 minutes	Beyond 5 minutes
Denormalized Table - 393216 Rows of data is table	0.09	0.05	0.05	0.05	0.05	0.058

	Run 1	Run 2	Run 3	Run 4	Run 5	Average
Normalized Table - 49152 Row of data is table	92.15	61.08	25.31	25.91	25.29	45.948
Denormalized Table - 49152 Rows of data is table	0.01	0.01	0.01	0.01	00.00 (less than 00.01)	0.008

Comments:

- The normalized system requires the use of joins to link employees to the shift they have worked as employee names are not stored in the shift table.
- Since the original table took more than 5 minutes for the normalized table query to run, we ran both queries on significantly smaller tables (more than 6 times smaller) and compared the readings.
- On the 5th run of the denormalized table query execution on the tables with “49152” rows (and a few other runs afterwards), the denormalized table query executed so fast that there was no reading to record, so the average is based on the first 4 runs.
- It is clear that the normalized tables' queries struggle to execute in a reasonable amount of time as the volume of data increases - while the denormalized table query continues to run to completion quickly even as the data increases exponentially.

Denormalization vs Normalization Test 2:
These queries look for the average amount of money that clients have paid for all the bookings that they have requested, and also find the minimum and the maximum amount of money they have paid for a booking.

Query for denormalized system:

```
/* DNORM QUERY 2 */
SELECT ClientID AS "Client", COUNT(*) AS "Total bookings",
AVG(AmountPaid) AS "Average booking cost", MAX(AmountPaid) AS "Maximum
paid", MIN(AmountPaid) AS "Minimum paid"
FROM BOOKING_DNORM_BIG
WHERE ClientID IS NOT NULL
GROUP BY ClientID;
```

Query for normalized system:

```
/* NORM QUERY 2 */
SELECT REGB.ClientID AS "Client", COUNT(*) AS "Total bookings",
AVG(PY.AmountPaid) AS "Average booking cost", MAX(PY.AmountPaid) AS
"Maximum paid", MIN(PY.AmountPaid) AS "Minimum paid"
FROM BOOKING_BIG BK
INNER JOIN RECURRENT_BOOKING_BIG RECB ON BK.BookingID = RECB.BookingID
INNER JOIN REGULAR_BOOKING_BIG REGB ON RECB.RegBookID = REGB.RegBookID
INNER JOIN PAYMENT_BIG PY ON PY.BookingID = BK.BookingID
GROUP BY ClientID;
```

	Run 1	Run 2	Run 3	Run 4	Run 5	Average
Normalized Table - 393216 Rows of data is table	Beyond 5 minutes	Beyond 5 minutes	Beyond 5 minutes	Beyond 5 minutes	Beyond 5 minutes	Beyond 5 minutes
Denormalized Table - 393216 Rows of data is table	0.11	0.10	0.10	0.10	00.10	0.102
Normalized Table - 6 Rows of data is table	00.00 (less than 00.01)	00.00 (less than 00.01)	00.00 (less than 00.01)	00.00 (less than 00.01)	00.00 (less than 00.01)	00.00 (less than 00.01)
Normalized Table - 96 Rows of data is table	0.67	0.67	0.67	0.67	0.67	0.682

Comments:

- Since the normalized table query required 3 joins to select the same data as the denormalized table query, executing it on a system with tables with around 400,000 rows of data respectively was likely too complicated for even the most capable of stand alone computer to compute, as the number of joins on tables that large would imply the select statement would search through many millions, if not, billions of rows of data.
- The Denormalized table query has no joins, so, once again, it executed quickly, as it has a smaller dataset to filter through.
- The normalized table query's time complexity was still significantly high, even with 96 rows in the table. This indicates that such a query would cause a lot of performance degradation on even a small or medium size (in terms of quantity of data) database system.

Denormalization vs Normalization Test 3:
The following query updates every record in the specified table with the specified field values. These statements allow us to assess one of the negative impacts of denormalization - that being the increased amount of processing time needed to update a denormalized table (since more fields need to be updated and assigned a value).

Query for denormalized system:

```
/* DNORM QUERY 3 */
UPDATE EMPLOYEE_DNORM_BIG
SET
            NINumber = rownum,
Address = '123 street',
PostCode = rownum,
City = 'LONDON',
FirstName = 'Name1',
LastName = 'Name2',
PPhoneNo = '01234567891',
SPhoneNo = '01234567891',
SEX = 'F',
DOB = TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
Email = 'example@email.com',
HireDate  = TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
DriverLicenseID = NULL,
PayType = NULL,
PayValue = NULL,
Salary = NULL,
TaxCode = NULL;
```

Query for normalized system:

```
/* NORM QUERY 3  */
UPDATE EMPLOYEE_BIG
SET
NINumber = rownum,
Address = '123 street',
PostCode = rownum,
FirstName = 'Name1',
LastName = 'Name2',
PPhoneNo = '01234567891',
SPhoneNo = '01234567891',
SEX = 'F',
DOB = TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
Email = 'example@email.com',
HireDate  = TO_DATE('02-NOV-2020', 'DD-MON-YYYY');
```

	Run 1	Run 2	Run 3	Run 4	Run 5	Average
Normalized Table - 40960 Rows of data is table	0.67	0.10	0.10	0.10	0.10	0.214
Denormalized Table - 40960 Rows of data is table	2.13	1.56	0.13	0.13	0.13	0.816

Comments:

- After caching, it seems like updating the denormalized table data only takes slightly longer than updating the normalized table data.
- Since you have to update more fields on a denormalized table and update more data on denormalized systems, this outcome is expected. While the system may take an additional amount of time to update the other tables in the normalized system which a denormalized table encompasses in a single table, fewer updates record would be required for relations that have a many to one relationship with the tables that would mainly be updated.

Indexing

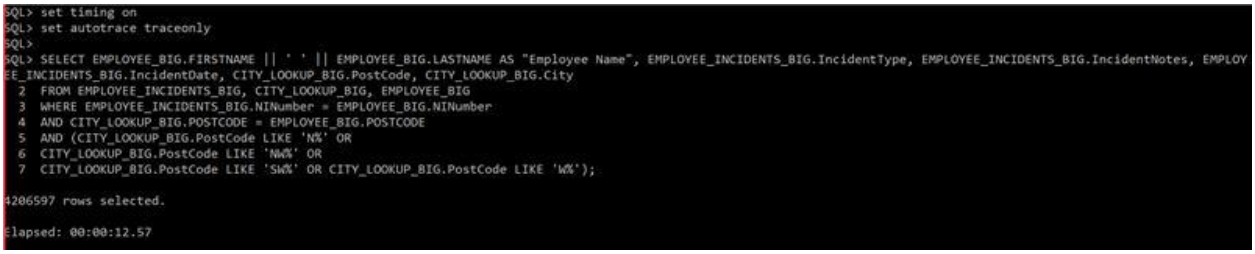
As it stands the queries used in our project use no indexes, so we decided to improve some selected queries using indexes. The first of these queries is:

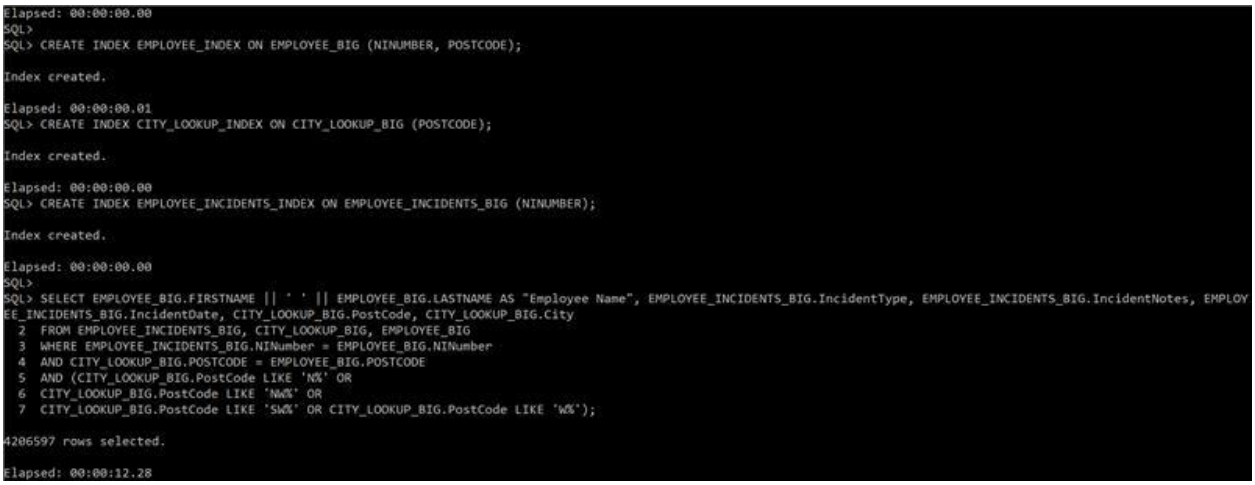
```
/* Query */
SELECT EMPLOYEE.FIRSTNAME || ' ' || EMPLOYEE.LASTNAME AS "Employee
Name", EMPLOYEE_INCIDENTS.IncidentType,
EMPLOYEE_INCIDENTS.IncidentNotes, EMPLOYEE_INCIDENTS.IncidentDate,
CITY_LOOKUP.PostCode, CITY_LOOKUP.City
FROM EMPLOYEE_INCIDENTS, CITY_LOOKUP, EMPLOYEE
WHERE EMPLOYEE_INCIDENTS.NINumber = EMPLOYEE.NINumber
AND CITY_LOOKUP.POSTCODE = EMPLOYEE.POSTCODE
AND (CITY_LOOKUP.PostCode LIKE 'N%' OR
CITY_LOOKUP.PostCode LIKE 'NW%' OR
CITY_LOOKUP.PostCode LIKE 'SW%' OR CITY_LOOKUP.PostCode LIKE 'W%');
```

To improve said query the following indexes were included:

```
CREATE INDEX EMPLOYEE_INDEX ON EMPLOYEE (NINUMBER, POSTCODE);
CREATE INDEX CITY_LOOKUP_INDEX ON CITY_LOOKUP (POSTCODE);
CREATE INDEX EMPLOYEE_INCIDENTS_INDEX ON EMPLOYEE_INCIDENTS (NINUMBER);
```

Below are screenshots of the first test of the query alongside the time it took for the query to take place without indexes (1st screenshot) and with them (2nd screenshot).





	TEST 1	TEST 2	TEST 3	TEST 4	TEST 5	AVERAGE TIME
Query Without Indexes	12.57	12.55	12.63	12.56	12.56	12.574
Query With Indexes	12.28	12.30	12.30	12.41	12.31	12.320
Difference in Time	0.29	0.25	0.33	0.15	0.25	0.254

The data shows a marginal increase in the performance of the sql statement as a result of the indexes. This may be due to the relatively small set of data (300000 rows). The performance increase was approximately 2%, which I imagine would be larger with much larger sets of data.

MongoDB Design

Employee collection

From the relational schema the table Employee would become a collection and the table City_Lookup would be removed. The columns that were used in the City_Lookup table would be replaced for an address document in the employee collection. Inside the employee collection, there would be an employee details document that would store data such as NINumber, FirstName, LastName, addresses, phone numbers, sex, DOB, email, shifts, incidents and HireDate. An address document would be embedded in the employee details document and will be one to one relationship between employee and an address. It will store data related to an address such as the postcode, street, city/town and country. Since the design contains a primary and secondary phone number, it will have a one to two relationship with the employee collection so there will be an array which will contain both phone numbers with the primary phone number stored into index 0 of the array and this will have a unique index created for it. The email field will also have a unique index created to ensure that emails are unique.

Employee and shift collection

The shift table in the relational schema would also be removed and replaced for a shift collection which is linked to the existing employee collection which will store all the shifts an employee has worked in a month. This will be split into two collections. The existing employee collection and the new shift collection. In the employee collection, because an employee can have zero or many shifts it will store information about each shift an employee has in an array of documents. This will include the shiftID, shiftDate and shiftType. The data will contain the shifts an employee has for the week. The shift collection will contain all the shifts an employee has worked, each shift will have a reference to the shift an employee has. The shift collection will store the shiftID, NINumber, ShiftDate and ShiftType. Each shift will contain a reference ID to the shift an employee has. This allows an employee to see the list of shifts they have for the week and the old shifts they have worked.

Employee and employee incidents collection

The Employee incidents table from the relational schema will be split into two collections, one being the existing employee collection and the other being the employee incident collection. Since an employee can have many incidents, it will be an array of documents which will store the IncidentID, IncidentType, IncidentNotes and the IncidentDate. Only the most recent incidents will be stored in the employee collection. The employee incident collection will store all the employee incidents, each incident will have a reference to the incident that the employee was involved in. The employee incident collection will store the IncidentID, NINumber, IncidentID, IncidentType, IncidentNotes and the IncidentDate.

Clients collection

There will be a collection called clients which will be used to store the details of clients similarly to how data will be stored in the employee collection. A document called client details would store the name, ClientID, ClientType, phone numbers, email and corporations. The phone number will have a unique index and since a client can have one to two phone numbers there will be an array which will store both phone numbers of the client. Since the email field will need to have a unique constraint, an index will be created for email so that it is unique. The Corporate_Client table from the relational schema will be removed and used

as a document. This will be an array embedded into client details so that the user can view lists of corporations a client belongs to.

Driver collection

The driver table in the relational schema will become a collection which will be used to store several data that is related to the driver. The Pay_Type table in the relational schema would be removed when migrating the design to MongoDB and the columns would be placed in the driver collection, this relates back to the concept that data which are generally required together in queries can be stored in the same collection. Since the driver collection contains information on details related to the driver, their pay details should be situated in the same location. A document called driver details will be stored in the driver collection which will contain the DriverID, EmployeeID, PayType, PayValue and cars. The EmployeeID will manually reference the NINumber in the employee collection. Since the driver can drive one or many cars, the cars document will be an array of documents embedded in the driver details document which would contain the car registration number, the registration year, mileage, LastMOTDate, number of seats, car status and ownerID. This would have the effect that you would be able to see the list of cars a driver operates and the details about them in the driver collection. The Driver_Revenue table and its columns in the relational schema would be translated into the driver collection in MongoDB and it will be a document embedded in the driver details document so that any information related to the driver needed for the job would all be stored in the driver collection. This would include the RevenueID, GrossTakings, PaidToCompany and NetEarnings. The registration number and employeeID will have a unique index.

Operator collection

There will be an operator collection in MongoDB which will be similar to the driver collection and will store details related to the operator job in the database. A document called operator details would store data such as the EmployeeID, salary and TaxCode. The EmployeeID will reference the NINumber in the employee collection and it will have a unique index.

Booking collection

The booking table in the relational schema will become a booking collection in MongoDB. In this collection it will contain a booking details document which will contain information on the OperatorID, DriverID, DateBooked, PickUpDate, PickUpAddress, DropOffAddress, NoOfPassengers and the customer phone number. Both the OperatorID and DriverID will be a reference to their respective collections and will have a unique index created so there are no duplicates. The payment table in the relational schema will be replaced and become a payment document which will be embedded in the booking details document and contain the payment information. This includes the PaymentID, payment method, payment date, payees name, card number, card expiry date, CVV and the amount paid. These values for the card will be null if the payment was made in cash and the PaymentID will be unique.

Regular_Booking collection

The regular_booking collection will contain details related to bookings that occur daily, weekly or monthly. A document called booking details will store the RegBook, ClientID, booking frequency, start date, noOfTransportees and duration. The clientID will reference the clients collection. The RegBookID and ClientID will have a unique index. The table recurrent_bookings from the relational schema would be replaced for a recurrent_bookings document which will be an array of documents embedded in booking details. This document will contain the BookingID which will be referencing the booking collection.

Appendix

Create and Insert Statements for original normalized design

```
/* DROP STATEMENTS */
DROP TABLE RECURRENT_BOOKING;
DROP TABLE PAYMENT;
DROP TABLE CORPORATE_CLIENT;
DROP TABLE REGULAR_BOOKING;
DROP TABLE CLIENTS;
DROP TABLE CARS;
DROP TABLE DRIVER_REVENUE;
DROP TABLE BOOKING;
DROP TABLE DRIVER;
DROP TABLE PAY_TYPE;
DROP TABLE OPERATOR;
DROP TABLE EMPLOYEE_INCIDENTS;
DROP TABLE SHIFT;
DROP TABLE EMPLOYEE;
DROP TABLE CITY_LOOKUP;

/* CITY_LOOKUP Table */
CREATE TABLE CITY_LOOKUP(
POSTCODE VARCHAR(8) PRIMARY KEY,
CITY VARCHAR(50) NOT NULL
);

/* CLIENTS Table */
CREATE TABLE CLIENTS (
CLIENTID CHAR(9) PRIMARY KEY CHECK(LENGTH(CLIENTID)=9),
CLIENTTYPE VARCHAR(15) NOT NULL CHECK(LENGTH(CLIENTTYPE)>6)
CHECK(CLIENTTYPE IN ('CORPORATE', 'PRIVATE')),
CONTACTFNAME VARCHAR(255) NOT NULL,
CONTACTLNAME VARCHAR(255) NOT NULL,
PPHONENO CHAR(11) NOT NULL UNIQUE,
SPHONENO CHAR(11) UNIQUE,
EMAIL VARCHAR2(400) UNIQUE CHECK(EMAIL LIKE '%_@_%') CHECK(EMAIL LIKE
'%.%')
);
```

```
/* PAY_TYPE Table */
CREATE TABLE PAY_TYPE (
PAYTYPE VARCHAR(9) PRIMARY KEY,
PAYVALUE NUMBER(9) NOT NULL
);

/* EMPLOYEE Table */
CREATE TABLE EMPLOYEE(
NINUMBER CHAR(9) PRIMARY KEY,
POSTCODE VARCHAR(20) NOT NULL,
ADDRESS VARCHAR2(255) NOT NULL,
FIRSTNAME VARCHAR(255) NOT NULL,
LASTNAME VARCHAR(255) NOT NULL,
PPHONENO CHAR(11) NOT NULL UNIQUE,
SPHONENO CHAR(11),
SEX CHAR(1) NOT NULL CHECK(SEX IN ('M', 'F')),
DOB DATE NOT NULL,
EMAIL VARCHAR2(255) NOT NULL UNIQUE CHECK(EMAIL LIKE '%_@_%' AND EMAIL
LIKE '%.%'),
HIREDATE DATE NOT NULL,
CONSTRAINT FK_POSTCODE FOREIGN KEY(POSTCODE) REFERENCES
CITY_LOOKUP(POSTCODE) ON DELETE CASCADE
);

/* OPERATOR Table */
CREATE TABLE OPERATOR (
OPERATORID CHAR(9) PRIMARY KEY CHECK(LENGTH(OPERATORID)=9),
EMPLOYEEID CHAR(9) NOT NULL UNIQUE CHECK(LENGTH(EMPLOYEEID)=9),
SALARY NUMBER(8,2) NOT NULL  CHECK(SALARY>0),
TAXCODE VARCHAR(20) NOT NULL,
CONSTRAINT FK_OPERATOR_EMPLOYEEID FOREIGN KEY (EMPLOYEEID) REFERENCES
EMPLOYEE(NINUMBER) ON DELETE CASCADE
);
```

```
/* DRIVER Table */
CREATE TABLE DRIVER (
DRIVERLICENSEID CHAR(9) PRIMARY KEY CHECK(LENGTH(DRIVERLICENSEID )=9),
EMPLOYEEID CHAR(9) NOT NULL UNIQUE CHECK(LENGTH(EMPLOYEEID)=9),
PAYTYPE VARCHAR (9) NOT NULL,
CONSTRAINT FK_DRIVER_EMPLOYEEID FOREIGN KEY (EMPLOYEEID) REFERENCES
EMPLOYEE(NINUMBER) ON DELETE CASCADE,
CONSTRAINT FK_DRIVER_PAYTYPE FOREIGN KEY (PAYTYPE) REFERENCES
PAY_TYPE(PAYTYPE)
);

/* BOOKING Table */
CREATE TABLE BOOKING (
BOOKINGID CHAR(9) PRIMARY KEY CHECK(LENGTH(BOOKINGID)=9),
OPERATORID CHAR(9) CHECK(LENGTH(OPERATORID)=9),
DRIVERID CHAR(9) CHECK(LENGTH(DRIVERID)=9),
DATEBOOKED DATE NOT NULL,
PICKUPDATE DATE NOT NULL,
PICKUPADDRESS VARCHAR(255)  NOT NULL,
DROPOFFADDRESS VARCHAR2(255) NOT NULL,
NOOFPASSENGERS NUMBER(2) NOT NULL CHECK(NOOFPASSENGERS>=1),
CUSTOMERPHONENUMBER CHAR(11) NOT NULL,
CONSTRAINT FK_BOOKING_OPERATORID FOREIGN KEY (OPERATORID) REFERENCES
OPERATOR(OPERATORID) ON DELETE SET NULL,
CONSTRAINT FK_BOOKING_DRIVERID FOREIGN KEY (DRIVERID) REFERENCES
DRIVER(DRIVERLICENSEID) ON DELETE SET NULL
);

/* REGULAR_BOOKING Table */
CREATE TABLE REGULAR_BOOKING(
REGBOOKID CHAR(9) CHECK(LENGTH(REGBOOKID)=9) CONSTRAINT
PK_REGULAR_BOOKING PRIMARY KEY,
CLIENTID CHAR(9) CHECK(LENGTH(CLIENTID)=9) NOT NULL,
BOOKINGFREQUENCY VARCHAR(50) NOT NULL,
STARTDATE DATE NOT NULL,
NOOFTRANSPORTEES NUMBER(2) CHECK(NOOFTRANSPORTEES>=1),
DURATION VARCHAR(10) NOT NULL,
CONSTRAINT FK_REGULAR_BOOKING_CLIENTID FOREIGN KEY(CLIENTID) REFERENCES
CLIENTS(CLIENTID) ON DELETE CASCADE
);
```

```
/* DRIVER_REVENUE Table */
CREATE TABLE DRIVER_REVENUE(
REVENUEID CHAR(9) CHECK(LENGTH(REVENUEID)=9) CONSTRAINT
PK_DRIVER_REVENUE PRIMARY KEY,
DRIVERID CHAR(9) CHECK(LENGTH(DRIVERID)=9),
REVENUEDATE DATE NOT NULL,
GROSSTAKINGS NUMBER(8,2) NOT NULL CHECK(GROSSTAKINGS>0),
PAIDTOCOMPANY NUMBER(8,2) NOT NULL CHECK(PAIDTOCOMPANY>0),
NETEARNINGS NUMBER(8,2) NOT NULL CHECK(NETEARNINGS>0),
CONSTRAINT FK_DRIVER_REVENUE FOREIGN KEY(DRIVERID) REFERENCES
DRIVER(DRIVERLICENSEID) ON DELETE SET NULL
);

/* CORPERATE_CLIENT Table */
CREATE TABLE CORPORATE_CLIENT (
CLIENTID CHAR(9) PRIMARY KEY CHECK(LENGTH(CLIENTID)=9),
CORPERATIONNAME VARCHAR(50) NOT NULL,
CONSTRAINT FK_CORPORATE_CLIENT_CLIENTID FOREIGN KEY(CLIENTID)
REFERENCES CLIENTS(CLIENTID) ON DELETE CASCADE
);

/* EMPLOYEE_INCIDENTS Table */
CREATE TABLE EMPLOYEE_INCIDENTS (
INCIDENTID CHAR(9) PRIMARY KEY CHECK(LENGTH(INCIDENTID)=9),
NINUMBER CHAR(9) CHECK(LENGTH(NINUMBER)=9),
INCIDENTTYPE VARCHAR2(20) NOT NULL,
INCIDENTNOTES VARCHAR2(4000),
INCIDENTDATE DATE NOT NULL,
CONSTRAINT FK_EMPLOYEE_INCIDENTS FOREIGN KEY (NINUMBER) REFERENCES
EMPLOYEE(NINUMBER) ON DELETE SET NULL
);
```

```
/* CARS Table */
CREATE TABLE CARS (
REGISTRATIONNO CHAR(7) PRIMARY KEY CHECK(LENGTH(REGISTRATIONNO)<=7),
DRIVERID CHAR(9) NOT NULL CHECK(LENGTH(DRIVERID)=9),
REGISTRATIONYEAR INTEGER NOT NULL,
MILEAGE VARCHAR(10) NOT NULL CHECK(MILEAGE>0),
LASTMOTDATE DATE NOT NULL,
NOOFSEATS VARCHAR(10) NOT NULL CHECK(NOOFSEATS>=4),
CARSTATUS VARCHAR(30) NOT NULL,
OWNERID VARCHAR(9) NOT NULL,
CONSTRAINT FK_CARS_DRIVERID FOREIGN KEY (DRIVERID) REFERENCES
DRIVER(DRIVERLICENSEID) ON DELETE CASCADE
);

/* SHIFT Table */
CREATE TABLE SHIFT (
SHIFTID CHAR(9) PRIMARY KEY CHECK(LENGTH(SHIFTID)=9),
EMPLOYEEID CHAR(9) CHECK(LENGTH(EMPLOYEEID)=9),
SHIFTDATE DATE NOT NULL,
SHIFTTYPE VARCHAR(20) NOT NULL,
CONSTRAINT FK_SHIFT_EMPLOYEEID FOREIGN KEY (EMPLOYEEID) REFERENCES
EMPLOYEE(NINUMBER) ON DELETE SET NULL
);

/* PAYMENT Table */
CREATE TABLE PAYMENT (
PAYMENTID CHAR(9) PRIMARY KEY CHECK(LENGTH(PAYMENTID)=9),
BOOKINGID CHAR(9) CHECK(LENGTH(BOOKINGID)=9),
PAYMENTMETHOD VARCHAR(10) NOT NULL,
PAYMENTDATE DATE NOT NULL,
PAYEESFIRSTNAME VARCHAR(255) NOT NULL,
PAYEESLASTNAME VARCHAR(255) NOT NULL,
CARDNUMBER VARCHAR(16),
EXPIRYDATE DATE,
CVV CHAR(3) CHECK(LENGTH(CVV)=3),
AMOUNTPAID NUMBER(8,2) NOT NULL CHECK(AMOUNTPAID>0),
CONSTRAINT FK_PAYMENT_BOOKINGID FOREIGN KEY (BOOKINGID) REFERENCES
BOOKING(BOOKINGID) ON DELETE SET NULL
);
```

```
/* RECURRENT_BOOKING Table */
CREATE TABLE RECURRENT_BOOKING (
BOOKINGID CHAR(9) PRIMARY KEY CHECK(LENGTH(BOOKINGID)=9),
REGBOOKID CHAR(9) NOT NULL CHECK(LENGTH(REGBOOKID)=9),
CONSTRAINT FK_RECURRENT_BOOKING_BOOKINGID FOREIGN KEY (BOOKINGID)
REFERENCES BOOKING(BOOKINGID) ON DELETE CASCADE,
CONSTRAINT FK_RECURRENT_BOOKING_REGBOOKID FOREIGN KEY (REGBOOKID)
REFERENCES REGULAR_BOOKING(REGBOOKID) ON DELETE CASCADE
);

/* CITY_LOOKUP Inserts */
INSERT INTO CITY_LOOKUP VALUES ('N14 4AU', 'NORTH LONDON');
INSERT INTO CITY_LOOKUP VALUES ('NW8 9ZL', 'NORTH WEST LONDON');
INSERT INTO CITY_LOOKUP VALUES ('SW17 6AF', 'SOUTH WEST LONDON');
INSERT INTO CITY_LOOKUP VALUES ('W10 4AD', 'WEST LONDON');
INSERT INTO CITY_LOOKUP VALUES ('EN5 7QZ', 'ENFIELD');
INSERT INTO CITY_LOOKUP VALUES ('L40 0AA', 'LIVERPOOL');
INSERT INTO CITY_LOOKUP VALUES ('M1 2ER', 'MANCHESTER');
INSERT INTO CITY_LOOKUP VALUES ('B1 1DA', 'BIRMINGHAM');

/* EMPLOYEE Inserts */
INSERT INTO EMPLOYEE VALUES ('EP123456A','B1 1DA','5 ALLINGTON
ROAD','AADI','BACCI','07000000000',NULL,'M',TO_DATE('23-JAN-2002','DD-
MON-YYYY'),'EMAIL1@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456B','B1 1DA','44 ALLINGTON
ROAD','AALAM','BABEL','07000000001','07000000002','M',TO_DATE('12-JAN-
1998','DD-MON-YYYY'),'EMAIL2@EMAIL.COM',TO_DATE('18-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456C','B1 1DA','39 ALLINGTON
ROAD','ABBA','BACCARI','07000000003',NULL,'M',TO_DATE('23-JAN-
2002','DD-MON-YYYY'),'EMAIL3@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456D','B1 1DA','60 ALLINGTON
ROAD','ABDULLAH','BACCHUS','07000000004',NULL,'M',TO_DATE('12-FEB-
2001','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456E','B1 1DA','90 ALLINGTON
ROAD','AARON','BACHMAN','07000000005',NULL,'M',TO_DATE('12-JAN-
1971','DD-MON-YYYY'),'EMAIL5@EMAIL.COM',TO_DATE('01-JUN-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456F','N14 4AU','63 KEYSE
ROAD','BAKI','CADIZ','07000000006','07000000007','M',TO_DATE('16-OCT-
```

```
1978','DD-MON-YYYY'),'EMAIL6@EMAIL.COM',TO_DATE('05-MAR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456G','N14 4AU','23 KEYSE
ROAD','BALDWIN','CADLE','07000000008',NULL,'M',TO_DATE('11-FEB-
1986','DD-MON-YYYY'),'EMAIL7@EMAIL.COM',TO_DATE('04-AUG-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456H','N14 4AU','17 KEYSE
ROAD','BABETTE','CADOTTE','07000000009',NULL,'F',TO_DATE('04-OCT-
1993','DD-MON-YYYY'),'EMAIL8@EMAIL.COM',TO_DATE('04-AUG-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456I','N14 4AU','72 KEYSE
ROAD','BAHULA','CADORETTE','07000000010',NULL,'F',TO_DATE('25-SEP-
2002','DD-MON-YYYY'),'EMAIL9@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456J','N14 4AU','88 KEYSE
ROAD','BARBRA','CADOGEN','07000000011','07000000012','F',TO_DATE('21-
MAR-1990','DD-MON-YYYY'),'EMAIL10@EMAIL.COM',TO_DATE('04-JUN-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456K','NW8 9ZL','57 DUNSTAN
ROAD','CADEN','DERWIN','07000000013',NULL,'M',TO_DATE('21-NOV-
1972','DD-MON-YYYY'),'EMAIL11@EMAIL.COM',TO_DATE('16-JUN-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456L','NW8 9ZL','75 DUNSTAN
ROAD','CALISSA','DESMOND','07000000014',NULL,'F',TO_DATE('29-OCT-
1981','DD-MON-YYYY'),'EMAIL12@EMAIL.COM',TO_DATE('16-MAR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456M','NW8 9ZL','12 DUNSTAN
ROAD','CAITLYN','DEXTER','07000000015','07000000018','F',TO_DATE('20-
AUG-2000','DD-MON-YYYY'),'EMAIL13@EMAIL.COM',TO_DATE('01-SEP-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456N','NW8 9ZL','93 DUNSTAN
ROAD','CAI','DILLON','07000000016',NULL,'M',TO_DATE('02-JAN-1975','DD-
MON-YYYY'),'EMAIL14@EMAIL.COM',TO_DATE('27-OCT-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456O','NW8 9ZL','73 DUNSTAN
ROAD','CAILYN','DIXIE','07000000017','07000000019','F',TO_DATE('17-DEC-
1973','DD-MON-YYYY'),'EMAIL15@EMAIL.COM',TO_DATE('06-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456P','M1 2ER','17 ADELAIDE
GROVE','DAGNA','EARWOOD','07000000018',NULL,'F',TO_DATE('19-DEC-
1972','DD-MON-YYYY'),'EMAIL16@EMAIL.COM',TO_DATE('30-OCT-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456Q','M1 2ER','12 ADELAIDE
GROVE','DAHLIA','EASTIN','07000000019',NULL,'F',TO_DATE('16-JAN-
```

```
1989','DD-MON-YYYY'),'EMAIL17@EMAIL.COM',TO_DATE('18-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456R','M1 2ER','14 ADELAIDE
GROVE','DALILA','EAKES','07000000020','07000000021','F',TO_DATE('11-
SEP-2002','DD-MON-YYYY'),'EMAIL18@EMAIL.COM',TO_DATE('07-NOV-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456S','M1 2ER','14 ADELAIDE
GROVE','DALLAS','EAKES','07000000021','07000000020','M',TO_DATE('09-
JUN-2002','DD-MON-YYYY'),'EMAIL19@EMAIL.COM',TO_DATE('07-NOV-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456T','M1 2ER','19 ADELAIDE
GROVE','DALIYA','EASTON','07000000022','07000000019','F',TO_DATE('2-
JAN-1977','DD-MON-YYYY'),'EMAIL20@EMAIL.COM',TO_DATE('07-NOV-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456U','SW17 6AF','32 ABBEY
MEWS','ED','FAHL','07000000023','07000000028','M',TO_DATE('28-NOV-
1985','DD-MON-YYYY'),'EMAIL21@EMAIL.COM',TO_DATE('21-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456V','SW17 6AF','48 ABBEY
MEWS','EDD','FADEL','07000000024',NULL,'M',TO_DATE('13-MAR-1975','DD-
MON-YYYY'),'EMAIL22@EMAIL.COM',TO_DATE('05-JUN-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456W','SW17 6AF','2 ABBEY
MEWS','EDDY','FLETCHER','07000000025',NULL,'M',TO_DATE('10-SEP-
1977','DD-MON-YYYY'),'EMAIL23@EMAIL.COM',TO_DATE('24-MAR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456X','SW17 6AF','50 ABBEY
MEWS','EDGAR','FITZ','07000000026',NULL,'M',TO_DATE('19-AUG-1980','DD-
MON-YYYY'),'EMAIL24@EMAIL.COM',TO_DATE('26-MAY-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456Z','SW17 6AF','61 ABBEY
MEWS','EDWARD','FLOYD','07000000027',NULL,'M',TO_DATE('20-JAN-
1980','DD-MON-YYYY'),'EMAIL25@EMAIL.COM',TO_DATE('31-MAR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457A','L40 0AA','43 OCEANA
CLOSE','FABIEN','GARNER','07000000028',NULL,'M',TO_DATE('05-AUG-
1974','DD-MON-YYYY'),'EMAIL26@EMAIL.COM',TO_DATE('08-JAN-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457B','L40 0AA','93 OCEANA
CLOSE','FIDEL','GARFIELD','07000000029',NULL,'M',TO_DATE('16-SEP-
1981','DD-MON-YYYY'),'EMAIL27@EMAIL.COM',TO_DATE('26-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457C','L40 0AA','90 OCEANA
CLOSE','FARRAH','GARRICK','07000000030','07000000033','M',TO_DATE('23-
```

```
OCT-1992','DD-MON-YYYY'),'EMAIL28@EMAIL.COM',TO_DATE('03-JUL-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457D','L40 0AA','95 OCEANA
CLOSE','FAYIZ','GEORGE','07000000031',NULL,'M',TO_DATE('07-JUN-
1978','DD-MON-YYYY'),'EMAIL29@EMAIL.COM',TO_DATE('01-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457E','L40 0AA','25 OCEANA
CLOSE','FERLIN','GORAN','07000000034',NULL,'M',TO_DATE('06-APR-
2000','DD-MON-YYYY'),'EMAIL30@EMAIL.COM',TO_DATE('24-JUN-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457F','W10 4AD','6 PARRY
STREET','GABBY','HALSEY','07000000035',NULL,'F',TO_DATE('29-DEC-
1974','DD-MON-YYYY'),'EMAIL31@EMAIL.COM',TO_DATE('11-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457G','W10 4AD','1 PARRY
STREET','GILDA','HAMILTON','07000000036',NULL,'F',TO_DATE('18-AUG-
2000','DD-MON-YYYY'),'EMAIL32@EMAIL.COM',TO_DATE('22-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457H','W10 4AD','46 PARRY
STREET','GALILEO','HANSON','07000000037',NULL,'M',TO_DATE('23-MAR-
1986','DD-MON-YYYY'),'EMAIL33@EMAIL.COM',TO_DATE('22-APR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457I','W10 4AD','81 PARRY
STREET','GARRY','HANSON','07000000038',NULL,'M',TO_DATE('18-NOV-
1976','DD-MON-YYYY'),'EMAIL34@EMAIL.COM',TO_DATE('23-APR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457J','W10 4AD','28 PARRY
STREET','GARON','HAYES','07000000039','07000000040','M',TO_DATE('16-
DEC-1999','DD-MON-YYYY'),'EMAIL35@EMAIL.COM',TO_DATE('03-AUG-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457K','EN5 7QZ','16 BRISTOW
ROAD','GARON','INGRAM','07000000041',NULL,'M',TO_DATE('04-NOV-
1970','DD-MON-YYYY'),'EMAIL36@EMAIL.COM',TO_DATE('03-AUG-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457L','EN5 7QZ','96 BRISTOW
ROAD','GARON','IRVING','07000000042','07000000043','M',TO_DATE('24-JUL-
1988','DD-MON-YYYY'),'EMAIL37@EMAIL.COM',TO_DATE('03-AUG-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457M','EN5 7QZ','58 BRISTOW
ROAD','GARON','IBACH','07000000044','07000000045','M',TO_DATE('11-OCT-
1977','DD-MON-YYYY'),'EMAIL38@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-
YYYY'));
```

```
INSERT INTO EMPLOYEE VALUES ('EP123457N','EN5 7QZ','86 BRISTOW
ROAD','GARON','IIDA','07000000046',NULL,'M',TO_DATE('22-OCT-1974','DD-
MON-YYYY'),'EMAIL39@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123457O','EN5 7QZ','48 BRISTOW
ROAD','GARON','IKEDA','07000000047',NULL,'M',TO_DATE('23-SEP-1991','DD-
MON-YYYY'),'EMAIL40@EMAIL.COM',TO_DATE('07-NOV-2020','DD-MON-YYYY'));



/* CLIENTS Inserts */
INSERT INTO CLIENTS VALUES
('CL123456A','PRIVATE','EDWARD','ELRIC','07500000000',NULL,'CLEMAIL1@EM
AIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456B','PRIVATE','HASHIRAMA','SENJU','07500000001',NULL,'CLEMAIL2
@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456C','PRIVATE','RINTAROU','OKABE','07500000002','07500000022','
CLEMAIL3@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456D','PRIVATE','MINATO','NAMIKAZE','07500000003',NULL,'CLEMAIL4
@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456E','PRIVATE','L.','LAWLIET','07500000004',NULL,'CLEMAIL5@EMAI
L.COM');
INSERT INTO CLIENTS VALUES
('CL123456F','PRIVATE','MADARA','UCHIHA','07500000005',NULL,'CLEMAIL6@E
MAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456G','PRIVATE','ITACHI','UCHIHA','07500000006',NULL,'CLEMAIL7@E
MAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456H','CORPORATE','ERZA','SCARLET','08000000000','08000000008','
CLBEMAIL8@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456I','PRIVATE','HACHIMAN','HIKIGAYA','07500000007',NULL,'CLEMAI
L9@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456J','CORPORATE','TODOROKI','SHOTO','08000000001',NULL,'CLBEMAI
L10@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456K','CORPORATE','BAKUGO','KATSUKI','08000000002',NULL,'CLBEMAI
L11@EMAIL.COM');
```

```
INSERT INTO CLIENTS VALUES
('CL123456L','CORPORATE','IZUKU','MIDORIYA','08000000003',NULL,'CLBEMAI
L12@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456M','PRIVATE','EREN','JAEGER','07500000008',NULL,'CLEMAIL13@EM
AIL.COM');
INSERT INTO CLIENTS VALUES ('CL123456N','PRIVATE','ACE','D.
PORTGAS','07500000009',NULL,'CLEMAIL14@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456O','PRIVATE','ROY','MUSTANG','07500000010',NULL,'CLEMAIL15@EM
AIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456P','PRIVATE','DAZAI','OSAMU','07500000011',NULL,'CLEMAIL16@EM
AIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456Q','PRIVATE','LEVI','ACKERMAN','07500000012',NULL,'CLEMAIL17@
EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456R','PRIVATE','ALPHONSE','ELRIC','07500000013','07500000023','
CLEMAIL18@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456S','PRIVATE','TOGA','HIMIKO','07500000014',NULL,'CLEMAIL19@EM
AIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456T','PRIVATE','KANEKI','KEN','07500000015',NULL,'CLEMAIL20@EMA
IL.COM');
INSERT INTO CLIENTS VALUES
('CL123456U','PRIVATE','KURISU','MAKISE','07500000016',NULL,'CLEMAIL21@
EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456V','CORPORATE','GOKU','SON','08000000004','08000000009','CLBE
MAIL22@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456W','PRIVATE','YUNO','GASAI','07500000017',NULL,'CLEMAIL23@EMA
IL.COM');
INSERT INTO CLIENTS VALUES
('CL123456X','PRIVATE','KAZUTO','KIRIGAYA','07500000018',NULL,'CLEMAIL2
4@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456Y','CORPORATE','MIKOTO','MISAKA','08000000005','08000000010',
'CLBEMAIL25@EMAIL.COM');
```

```
INSERT INTO CLIENTS VALUES
('CL123456Z','CORPORATE','GINTOKI','SAKATA','08000000006','08000000011'
,'CLBEMAIL26@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123457A','PRIVATE','HISOKA','MOROW','07500000019','07500000024','CL
EMAIL27@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123457B','PRIVATE','LELOUCHE','LAMPEROUGE','07500000020',NULL,'CLEM
AIL28@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123457C','PRIVATE','ZORO','RORONOA','07500000021',NULL,'CLEMAIL29@E
MAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123457D','CORPORATE','KILLUA','ZOLDYCK','08000000007','08000000012'
,'CLBEMAIL30@EMAIL.COM');

/* PAY_TYPE Inserts */
INSERT INTO PAY_TYPE VALUES ('FIXEDTEMP', 24000);
INSERT INTO PAY_TYPE VALUES ('FIXEDFULL', 36000);
INSERT INTO PAY_TYPE VALUES ('COMM75', 75);
INSERT INTO PAY_TYPE VALUES ('COMM80', 80);
INSERT INTO PAY_TYPE VALUES ('COMM70', 70);
/* OPERATOR Inserts */
INSERT INTO OPERATOR VALUES('OP123456A','EP123456A',40000,'1250L');
INSERT INTO OPERATOR VALUES('OP123456B','EP123456B',42500,'1250L');
INSERT INTO OPERATOR VALUES('OP123456C','EP123456C',39000,'1250L');
INSERT INTO OPERATOR VALUES('OP123456D','EP123456D',36000,'1250L');
INSERT INTO OPERATOR VALUES('OP123456E','EP123456E',41000,'1250L');
INSERT INTO OPERATOR VALUES('OP123456F','EP123456F',41000,'1250L');
INSERT INTO OPERATOR VALUES('OP123456G','EP123456G',40000,'1250L');
INSERT INTO OPERATOR VALUES('OP123456H','EP123456H',38000,'1250L');

/* DRIVER Inserts */
INSERT INTO DRIVER VALUES ('DV123456A','EP123456I','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456B','EP123456J','COMM75');
INSERT INTO DRIVER VALUES ('DV123456C','EP123456K','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456D','EP123456L','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456E','EP123456M','COMM75');
INSERT INTO DRIVER VALUES ('DV123456F','EP123456N','COMM80');
INSERT INTO DRIVER VALUES ('DV123456G','EP123456O','COMM75');
INSERT INTO DRIVER VALUES ('DV123456H','EP123456P','COMM75');
INSERT INTO DRIVER VALUES ('DV123456I','EP123456Q','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456J','EP123456R','COMM75');
```

```
INSERT INTO DRIVER VALUES ('DV123456K','EP123456S','FIXEDTEMP');
INSERT INTO DRIVER VALUES ('DV123456L','EP123456T','COMM75');
INSERT INTO DRIVER VALUES ('DV123456M','EP123456U','FIXEDTEMP');
INSERT INTO DRIVER VALUES ('DV123456N','EP123456V','COMM75');
INSERT INTO DRIVER VALUES ('DV123456O','EP123456W','COMM75');
INSERT INTO DRIVER VALUES ('DV123456P','EP123456X','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456R','EP123456Z','FIXEDTEMP');
INSERT INTO DRIVER VALUES ('DV123456S','EP123457A','FIXEDTEMP');
INSERT INTO DRIVER VALUES ('DV123456T','EP123457B','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456U','EP123457C','COMM80');
INSERT INTO DRIVER VALUES ('DV123456V','EP123457D','COMM75');
INSERT INTO DRIVER VALUES ('DV123456W','EP123457E','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456X','EP123457F','COMM75');
INSERT INTO DRIVER VALUES ('DV123456Y','EP123457G','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456Z','EP123457H','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123457A','EP123457I','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123457B','EP123457J','COMM75');
INSERT INTO DRIVER VALUES ('DV123457C','EP123457K','FIXEDTEMP');
INSERT INTO DRIVER VALUES ('DV123457D','EP123457L','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123457E','EP123457M','COMM70');
INSERT INTO DRIVER VALUES ('DV123457F','EP123457N','COMM75');
INSERT INTO DRIVER VALUES ('DV123457G','EP123457O','FIXEDFULL');

/* BOOKING Inserts */
INSERT INTO BOOKING VALUES
('BK123456A','OP123456A','DV123456B',TO_DATE('08-NOV-2020','DD-MON-
YYYY'),TO_DATE('12-NOV-2020','DD-MON-YYYY'),'21 BALFE STREET','79
TEMPLE WAY',3,'07900000000');
INSERT INTO BOOKING VALUES
('BK123456B','OP123456B','DV123456B',TO_DATE('07-MAY-2020','DD-MON-
YYYY'),TO_DATE('08-MAY-2020','DD-MON-YYYY'),'78 TRAILLE STREET','76
BOAR LANE',5,'07900000001');
INSERT INTO BOOKING VALUES
('BK123456C','OP123456G','DV123456X',TO_DATE('15-APR-2020','DD-MON-
YYYY'),TO_DATE('15-APR-2020','DD-MON-YYYY'),'72 QUAY STREET','63 PRINCE
CONSORT ROAD',5,'07900000002');
INSERT INTO BOOKING VALUES
('BK123456D','OP123456C','DV123456W',TO_DATE('18-MAY-2020','DD-MON-
YYYY'),TO_DATE('18-MAY-2020','DD-MON-YYYY'),'49 MILL LANE','77 HUDSTON
ST',7,'07900000003');
INSERT INTO BOOKING VALUES
('BK123456E','OP123456D','DV123456A',TO_DATE('07-OCT-2020','DD-MON-
```

```
YYYY'),TO_DATE('09-OCT-2020','DD-MON-YYYY'),'49 LONG STREET','118
TERRICK RD',4,'07900000004');
INSERT INTO BOOKING VALUES
('BK123456F','OP123456F','DV123457A',TO_DATE('10-SEP-2020','DD-MON-
YYYY'),TO_DATE('10-SEP-2020','DD-MON-YYYY'),'91 NITH STREET','14 CHAPEL
LANE',1,'07900000005');
INSERT INTO BOOKING VALUES
('BK123456G','OP123456E','DV123456G',TO_DATE('09-JUN-2020','DD-MON-
YYYY'),TO_DATE('21-JUN-2020','DD-MON-YYYY'),'43 BOUVERIE ROAD','95
CARRIERS ROAD',4,'07900000006');
INSERT INTO BOOKING VALUES
('BK123456H','OP123456G','DV123456K',TO_DATE('18-AUG-2020','DD-MON-
YYYY'),TO_DATE('16-SEP-2020','DD-MON-YYYY'),'72 HENDFORD HILL','94
BRYNGLAS ROAD',2,'07900000007');
INSERT INTO BOOKING VALUES
('BK123456I','OP123456H','DV123456O',TO_DATE('03-SEP-2020','DD-MON-
YYYY'),TO_DATE('13-SEP-2020','DD-MON-YYYY'),'69 OXFORD RD','109
HARROGATE ROAD',1,'07900000008');
INSERT INTO BOOKING VALUES
('BK123456J','OP123456E','DV123456M',TO_DATE('23-JUL-2020','DD-MON-
YYYY'),TO_DATE('25-JUL-2020','DD-MON-YYYY'),'60 MALBOROUGH
CRESCENT','72 BUCKINGHAM RD',1,'07900000009');
INSERT INTO BOOKING VALUES
('BK123456K','OP123456C','DV123456H',TO_DATE('09-JUN-2020','DD-MON-
YYYY'),TO_DATE('25-JUN-2020','DD-MON-YYYY'),'38 PRESTWICK ROAD','138
HULL ROAD',1,'07900000010');
INSERT INTO BOOKING VALUES
('BK123456L','OP123456D','DV123456A',TO_DATE('12-SEP-2020','DD-MON-
YYYY'),TO_DATE('25-SEP-2020','DD-MON-YYYY'),'88 HELLAND BRIDGE','102
STATION ROAD',1,'07900000011');
INSERT INTO BOOKING VALUES
('BK123456M','OP123456A','DV123456T',TO_DATE('13-AUG-2020','DD-MON-
YYYY'),TO_DATE('25-AUG-2020','DD-MON-YYYY'),'80 PARK AVENUE','103
CUNNERY ROAD',5,'07900000012');
INSERT INTO BOOKING VALUES
('BK123456N','OP123456F','DV123456Z',TO_DATE('13-MAY-2020','DD-MON-
YYYY'),TO_DATE('07-JUN-2020','DD-MON-YYYY'),'55 COAST RD','36 HIGH
STREET',1,'07900000013');
INSERT INTO BOOKING VALUES
('BK123456O','OP123456B','DV123456K',TO_DATE('12-AUG-2020','DD-MON-
YYYY'),TO_DATE('12-AUG-2020','DD-MON-YYYY'),'60 GOLDEN KNOWES ROAD','57
NORTH PROMENADE',6,'07900000014');
```

```
INSERT INTO BOOKING VALUES
('BK123456P','OP123456C','DV123456P',TO_DATE('24-JUL-2020','DD-MON-
YYYY'),TO_DATE('25-JUL-2020','DD-MON-YYYY'),'111 CAMBRIDGE ROAD','84
VICTORIA ROAD',2,'07900000015');
INSERT INTO BOOKING VALUES
('BK123456Q','OP123456G','DV123457F',TO_DATE('23-AUG-2020','DD-MON-
YYYY'),TO_DATE('23-AUG-2020','DD-MON-YYYY'),'27 STONE CELLAR ROADT','54
FORE ST',1,'07900000016');
INSERT INTO BOOKING VALUES
('BK123456R','OP123456F','DV123456Z',TO_DATE('24-SEP-2020','DD-MON-
YYYY'),TO_DATE('24-SEP-2020','DD-MON-YYYY'),'39 GEORGE STREET','100
BOTLEY ROAD',1,'07900000017');
INSERT INTO BOOKING VALUES
('BK123456S','OP123456A','DV123456E',TO_DATE('23-OCT-2020','DD-MON-
YYYY'),TO_DATE('23-OCT-2020','DD-MON-YYYY'),'59 EXNING ROAD','105
WALDEN ROAD',7,'07900000018');
INSERT INTO BOOKING VALUES
('BK123456T','OP123456H','DV123456O',TO_DATE('23-OCT-2020','DD-MON-
YYYY'),TO_DATE('23-OCT-2020','DD-MON-YYYY'),'71 HOLGATE RD','102
BRACKLEY ROAD',1,'07900000019');

/* REGULAR_BOOKING Inserts */
INSERT INTO REGULAR_BOOKING VALUES
('RB123456A','CL123456X','WEEKLY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),3,'7 WEEKS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456B','CL123456U','DAILY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),5,'4 DAYS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456C','CL123456I','WEEKLY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),5,'4 WEEKS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456D','CL123456G','DAILY',TO_DATE('01-NOV-2020','DD-MON-
YYYY'),7,'2 DAYS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456E','CL123457D','DAILY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),4,'30 DAYS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456F','CL123456Z','WEEKLY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),1,'12 WEEKS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456G','CL123457C','WEEKLY',TO_DATE('01-NOV-2020','DD-MON-
YYYY'),2,'6 WEEKS');
```

```
INSERT INTO REGULAR_BOOKING VALUES
('RB123456H','CL123456C','DAILY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),1,'3 DAYS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456I','CL123456A','WEEKLY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),1,'8 WEEKS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456J','CL123456Q','DAILY',TO_DATE('01-NOV-2020','DD-MON-
YYYY'),1,'7 DAYS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456K','CL123456E','MONTHLY',TO_DATE('06-NOV-2020','DD-MON-
YYYY'),1,'6 MONTHS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456L','CL123457B','WEEKLY',TO_DATE('06-OCT-2020','DD-MON-
YYYY'),6,'6 WEEKS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456M','CL123456X','MONTHLY',TO_DATE('22-OCT-2020','DD-MON-
YYYY'),2,'4 MONTHS');
INSERT INTO REGULAR_BOOKING VALUES
('RB123456N','CL123456J','DAILY',TO_DATE('14-OCT-2020','DD-MON-
YYYY'),1,'14 DAYS');
INSERT INTO REGULAR_BOOKING VALUES
('RB1234560','CL123456L','DAILY',TO_DATE('12-OCT-2020','DD-MON-
YYYY'),7,'12 DAYS');

/* DRIVER_REVENUE Inserts */
INSERT INTO DRIVER_REVENUE VALUES ('RV123456A','DV123456A',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,476,2524);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456B','DV123456B',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2932,833,2099);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456C','DV123456C',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,497,2503);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456D','DV123456D',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,320,2680);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456E','DV123456E',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3265,916.25,2348.75);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456F','DV123456F',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3004,700.8,2303.2);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456G','DV123456G',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2761,690.25,2070.75);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456H','DV123456H',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2637,759.25,1877.75);
```

```
INSERT INTO DRIVER_REVENUE VALUES ('RV123456I','DV123456I',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,306,2694);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456J','DV123456J',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3222,905.5,2316.5);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456K','DV123456K',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2000,467,1533);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456L','DV123456L',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2765,791.25,1973.75);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456M','DV123456M',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2000,468,1532);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456N','DV123456N',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2675,768.75,1906.25);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456O','DV123456O',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2572,743,1829);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456P','DV123456P',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,405,2595);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456Q','DV123456R',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2000,342,1658);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456R','DV123456S',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2000,470,1530);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456S','DV123456T',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,317,2683);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456T','DV123456U',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3035,707,2328);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456U','DV123456V',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2861,672.20,2188.8);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456V','DV123456W',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,395,2605);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456W','DV123456X',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3303,925.75,2377.25);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456X','DV123456Y',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,475,2525);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456Y','DV123456Z',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,420,2580);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456Z','DV123457A',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,308,2692);
INSERT INTO DRIVER_REVENUE VALUES ('RV123457A','DV123457B',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3319,929.75,2389.25);
INSERT INTO DRIVER_REVENUE VALUES ('RV123457B','DV123457C',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2000,288,1712);
INSERT INTO DRIVER_REVENUE VALUES ('RV123457C','DV123457D',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,321,2679);
```

```
INSERT INTO DRIVER_REVENUE VALUES ('RV123457D','DV123457E',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2506,851.8,1654.2);
INSERT INTO DRIVER_REVENUE VALUES ('RV123457E','DV123457F',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2623,755.75,1867.25);
INSERT INTO DRIVER_REVENUE VALUES ('RV123457F','DV123457G',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,434,2566);

/* CORPERATE_CLIENT Inserts */
INSERT INTO CORPORATE_CLIENT VALUES('CL123456H','UMBRELLA CORP');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456J','WONKA INDUSTRIES');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456K','STARK INDUSTRIES');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456L','WAYNE ENTERPRISES');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456V','INGEN');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456Y','SPECTRE');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456Z','MONSTERS INC');
INSERT INTO CORPORATE_CLIENT VALUES('CL123457D','THE KRUSTY KRAB');



/* EMPLOYEE_INCIDENTS Inserts*/
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456A','EP123457I','SICK','GOT MY FINGERS STUCK IN A BOWLING
BALL',TO_DATE('02-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456B','EP123456M','VACATION REQUEST','I WANT TO GO ON MY
HONEYMOON',TO_DATE('10-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456C','EP123456G','DISCIPLINARY ACTION','CAME TO WORK DRUNK.
LICENSE SUSPENDED UNTIL FURTHER NOTICE',TO_DATE('02-OCT-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456D','EP123456X','SICK','MY GIRLFRIEND BIT ME IN A BAD
PLACE',TO_DATE('12-OCT-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456E','EP123457I','SICK','I HAVE A BLOCKED NOSE',TO_DATE('15-
MAY-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456F','EP123457B','SICK','BITTEN BY A SNAKE',TO_DATE('28-APR-
2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456G','EP123457K','VACATION REQUEST','LONG OVERDUE FOR A BREAK,
WANT TO TAKE MY FAMILY TO HAWAII',TO_DATE('03-SEP-2020','DD-MON-
YYYY'));
```

```
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456H','EP123456F','VACATION REQUEST','I HEAR SPAIN IS A GREAT
PLACE TO FIND CARS, MIND IF I TAKE A LOOK MYSELF?',TO_DATE('07-MAR-
2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456I','EP123457A','SICK','REALLY BAD HANGOVER',TO_DATE('09-APR-
2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456J','EP123456Z','SICK','DRANK TOILET WATER, I DONT FEEL SO
GOOD',TO_DATE('03-NOV-2020','DD-MON-YYYY'));

/* CARS Inserts */
INSERT INTO CARS VALUES('CR1234A','DV123456A',2018,52041,TO_DATE('06-
AUG-2020','DD-MON-YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS VALUES('CR1234B','DV123456B',2008,112218,TO_DATE('28-
AUG-2020','DD-MON-YYYY'),4,'AWAITING REPAIR','COMPANY*');
INSERT INTO CARS VALUES('CR1234C','DV123456C',2012,178627,TO_DATE('28-
APR-2020','DD-MON-YYYY'),5,'ROADWORTHY','DV123456C');
INSERT INTO CARS VALUES('CR1234D','DV123456D',2011,234439,TO_DATE('28-
JUL-2020','DD-MON-YYYY'),4,'IN FOR SERVICE','DV123456D');
INSERT INTO CARS VALUES('CR1234E','DV123456E',2016,138258,TO_DATE('05-
AUG-2020','DD-MON-YYYY'),7,'ROADWORTHY','DV123456E');
INSERT INTO CARS VALUES('CR1234F','DV123456F',2013,198394,TO_DATE('04-
JUL-2020','DD-MON-YYYY'),4,'ROADWORTHY','DV123456F');
INSERT INTO CARS VALUES('CR1234G','DV123456G',2010,204617,TO_DATE('11-
JUL-2020','DD-MON-YYYY'),4,'IN FOR SERVICE','DV123456G');
INSERT INTO CARS VALUES('CR1234H','DV123456H',2014,145298,TO_DATE('15-
DEC-2019','DD-MON-YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS VALUES('CR1234I','DV123456I',2011,144342,TO_DATE('26-
FEB-2020','DD-MON-YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS VALUES('CR1234J','DV123456J',2010,76744,TO_DATE('06-
MAR-2020','DD-MON-YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS VALUES('CR1234K','DV123456K',2017,152576,TO_DATE('05-
FEB-2020','DD-MON-YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS VALUES('CR1234L','DV123456L',2013,57314,TO_DATE('09-
JUL-2020','DD-MON-YYYY'),4,'ROADWORTHY','DV123456L');
INSERT INTO CARS VALUES('CR1234M','DV123456M',2018,162746,TO_DATE('19-
APR-2020','DD-MON-YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS VALUES('CR1234N','DV123456N',2017,76487,TO_DATE('25-
SEP-2020','DD-MON-YYYY'),4,'AWAITING REPAIR','COMPANY*');
INSERT INTO CARS VALUES('CR1234O','DV123456O',2009,74797,TO_DATE('01-
AUG-2020','DD-MON-YYYY'),4,'ROADWORTHY','DV123456O');
```

```
INSERT INTO CARS VALUES('CR1234P','DV123456P',2010,192497,TO_DATE('18-
DEC-2019','DD-MON-YYYY'),4,'ROADWORTHY','DV123456P');
INSERT INTO CARS VALUES('CR1234Q','DV123456R',2013,247143,TO_DATE('31-
AUG-2020','DD-MON-YYYY'),5,'IN FOR SERVICE','DV123456R');
INSERT INTO CARS VALUES('CR1234R','DV123456S',2013,162729,TO_DATE('23-
JAN-2020','DD-MON-YYYY'),5,'ROADWORTHY','DV123456S');
INSERT INTO CARS VALUES('CR1234S','DV123456T',2014,205776,TO_DATE('08-
APR-2020','DD-MON-YYYY'),4,'IN FOR SERVICE','DV123456T');
INSERT INTO CARS VALUES('CR1234T','DV123456U',2017,232771,TO_DATE('05-
AUG-2020','DD-MON-YYYY'),4,'IN FOR SERVICE','COMPANY*');
INSERT INTO CARS VALUES('CR1234U','DV123456V',2009,230137,TO_DATE('15-
JAN-2020','DD-MON-YYYY'),4,'IN FOR SERVICE','COMPANY*');
INSERT INTO CARS VALUES('CR1234V','DV123456W',2015,175932,TO_DATE('26-
OCT-2020','DD-MON-YYYY'),4,'ROADWORTHY','DV123456W');
INSERT INTO CARS VALUES('CR1234W','DV123456X',2012,230268,TO_DATE('07-
OCT-2020','DD-MON-YYYY'),4,'IN FOR SERVICE','DV123456X');
INSERT INTO CARS VALUES('CR1234X','DV123456Y',2007,170101,TO_DATE('20-
SEP-2020','DD-MON-YYYY'),4,'ROADWORTHY','DV123456Y');
INSERT INTO CARS VALUES('CR1234Y','DV123456Z',2012,226776,TO_DATE('14-
JUL-2020','DD-MON-YYYY'),4,'IN FOR SERVICE','DV123456Z');
INSERT INTO CARS VALUES('CR1234Z','DV123457A',2014,73036,TO_DATE('23-
AUG-2020','DD-MON-YYYY'),6,'ROADWORTHY','DV123457A');
INSERT INTO CARS VALUES('CR1235A','DV123457B',2017,202035,TO_DATE('27-
AUG-2020','DD-MON-YYYY'),4,'ROADWORTHY','DV123457B');
INSERT INTO CARS VALUES('CR1235B','DV123457C',2008,120144,TO_DATE('02-
JAN-2020','DD-MON-YYYY'),4,'AWAITING REPAIR','DV123457C');
INSERT INTO CARS VALUES('CR1235C','DV123457D',2006,166016,TO_DATE('14-
SEP-2020','DD-MON-YYYY'),6,'ROADWORTHY','DV123457D');
INSERT INTO CARS VALUES('CR1235D','DV123457E',2012,87452,TO_DATE('01-
APR-2020','DD-MON-YYYY'),4,'ROADWORTHY','DV123457E');
INSERT INTO CARS VALUES('CR1235E','DV123457F',2013,88800,TO_DATE('01-
NOV-2020','DD-MON-YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS VALUES('CR1235F','DV123457G',2012,130650,TO_DATE('17-
AUG-2020','DD-MON-YYYY'),6,'AWAITING REPAIR','DV123457G');

/* SHIFT Inserts */
INSERT INTO SHIFT VALUES ('SH123456A','EP123456V',TO_DATE('05-NOV-
2020','DD-MON-YYYY'),'NIGHT');
INSERT INTO SHIFT VALUES ('SH123456B','EP123456Z',TO_DATE('25-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456C','EP123457I',TO_DATE('29-NOV-
2020','DD-MON-YYYY'),'NIGHT');
```

```
INSERT INTO SHIFT VALUES ('SH123456D','EP123457K',TO_DATE('05-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456E','EP123457N',TO_DATE('13-NOV-
2020','DD-MON-YYYY'),'EVENING');
INSERT INTO SHIFT VALUES ('SH123456F','EP123456Z',TO_DATE('08-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456G','EP123457A',TO_DATE('24-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456H','EP123456W',TO_DATE('28-NOV-
2020','DD-MON-YYYY'),'NIGHT');
INSERT INTO SHIFT VALUES ('SH123456I','EP123456B',TO_DATE('12-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456J','EP123456I',TO_DATE('24-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456K','EP123457E',TO_DATE('04-NOV-
2020','DD-MON-YYYY'),'EVENING');
INSERT INTO SHIFT VALUES ('SH123456L','EP123457D',TO_DATE('29-NOV-
2020','DD-MON-YYYY'),'EVENING');
INSERT INTO SHIFT VALUES ('SH123456M','EP123457M',TO_DATE('07-NOV-
2020','DD-MON-YYYY'),'EVENING');
INSERT INTO SHIFT VALUES ('SH123456N','EP123456K',TO_DATE('07-NOV-
2020','DD-MON-YYYY'),'EVENING');
INSERT INTO SHIFT VALUES ('SH1234560','EP123456N',TO_DATE('11-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456P','EP123456X',TO_DATE('11-NOV-
2020','DD-MON-YYYY'),'NIGHT');
INSERT INTO SHIFT VALUES ('SH123456Q','EP123457I',TO_DATE('10-NOV-
2020','DD-MON-YYYY'),'NIGHT');
INSERT INTO SHIFT VALUES ('SH123456R','EP123456T',TO_DATE('28-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456S','EP123457C',TO_DATE('07-NOV-
2020','DD-MON-YYYY'),'EVENING');
INSERT INTO SHIFT VALUES ('SH123456T','EP123456D',TO_DATE('02-NOV-
2020','DD-MON-YYYY'),'NIGHT');

/* PAYMENT Inserts */
INSERT INTO PAYMENT VALUES('PY123456A','BK123456A','CARD',TO_DATE('08-
NOV-2020','DD-MON-
YYYY'),'KAZUTO','KIRIGAYA','4432491216876558',TO_DATE('AUG-2023','MON-
YYYY'),'296',14.43);
INSERT INTO PAYMENT VALUES('PY123456B','BK123456B','CARD',TO_DATE('07-
MAY-2020','DD-MON-
```

```
YYYY'),'KURISU','MAKISE','5127071833675349',TO_DATE('AUG-2023','MON-
YYYY'),'314',39.3);
INSERT INTO PAYMENT VALUES('PY123456C','BK123456C','CARD',TO_DATE('15-
APR-2020','DD-MON-
YYYY'),'HACHIMAN','HIKIGAYA','5213211675421049',TO_DATE('AUG-
2023','MON-YYYY'),'329',37.9);
INSERT INTO PAYMENT VALUES('PY123456D','BK123456D','CARD',TO_DATE('18-
MAY-2020','DD-MON-
YYYY'),'ITACHI','UCHIHA','4415655174552366',TO_DATE('AUG-2023','MON-
YYYY'),'129',51.66);
INSERT INTO PAYMENT VALUES('PY123456E','BK123456E','CARD',TO_DATE('07-
OCT-2020','DD-MON-
YYYY'),'KILLUA','ZOLDYCK','5416855454432331',TO_DATE('AUG-2023','MON-
YYYY'),'897',29.08);
INSERT INTO PAYMENT VALUES('PY123456F','BK123456F','CARD',TO_DATE('10-
SEP-2020','DD-MON-
YYYY'),'SAKATA','GINTOKI','5177185476318618',TO_DATE('AUG-2023','MON-
YYYY'),'316',4.54);
INSERT INTO PAYMENT VALUES('PY123456G','BK123456G','CASH',TO_DATE('21-
JUN-2020','DD-MON-YYYY'),'NARUTO','UZUMAKI',NULL,NULL,NULL,26.16);
INSERT INTO PAYMENT VALUES('PY123456H','BK123456H','CARD',TO_DATE('18-
AUG-2020','DD-MON-
YYYY'),'RORONOA','ZORO','4966118739314742',TO_DATE('AUG-2022','MON-
YYYY'),'640',14.06);
INSERT INTO PAYMENT VALUES('PY123456I','BK123456I','CARD',TO_DATE('03-
SEP-2020','DD-MON-
YYYY'),'RINTAROU','OKABE','5275496342325815',TO_DATE('DEC-2020','MON-
YYYY'),'664',5.42);
INSERT INTO PAYMENT VALUES('PY123456J','BK123456J','CARD',TO_DATE('23-
JUL-2020','DD-MON-
YYYY'),'LIGHT','YAGAMI','5258908674928124',TO_DATE('DEC-2020','MON-
YYYY'),'366',4.99);
INSERT INTO PAYMENT VALUES('PY123456K','BK123456K','CARD',TO_DATE('09-
JUN-2020','DD-MON-
YYYY'),'EDWARD','ELRIC','5420754499865718',TO_DATE('AUG-2022','MON-
YYYY'),'532',7.65);
INSERT INTO PAYMENT VALUES('PY123456L','BK123456L','CARD',TO_DATE('12-
SEP-2020','DD-MON-
YYYY'),'LEVI','ACKERMAN','4985301649478610',TO_DATE('DEC-2020','MON-
YYYY'),'528',5.04);
INSERT INTO PAYMENT VALUES('PY123456M','BK123456M','CARD',TO_DATE('13-
AUG-2020','DD-MON-YYYY'),'LUFFY','MONKEY
D.','5456334646753146',TO_DATE('AUG-2022','MON-YYYY'),'275',38.9);
```

```
INSERT INTO PAYMENT VALUES('PY123456N','BK123456N','CARD',TO_DATE('13-
MAY-2020','DD-MON-
YYYY'),'L.','LAWLIET','5136226533936057',TO_DATE('AUG-2022','MON-
YYYY'),'177',4.17);
INSERT INTO PAYMENT VALUES('PY123456O','BK123456O','CARD',TO_DATE('12-
AUG-2020','DD-MON-
YYYY'),'LELOUCHE','LAMPEROUGE','5445471841716637',TO_DATE('AUG-
2021','MON-YYYY'),'314',39.84);
INSERT INTO PAYMENT VALUES('PY123456P','BK123456P','CASH',TO_DATE('25-
JUL-2020','DD-MON-YYYY'),'KAZUTO','KIRIGAYA',NULL,NULL,NULL,15.46);
INSERT INTO PAYMENT VALUES('PY123456Q','BK123456Q','CASH',TO_DATE('23-
AUG-2020','DD-MON-YYYY'),'GON','FREECSS',NULL,NULL,NULL,4.38);
INSERT INTO PAYMENT VALUES('PY123456R','BK123456R','CARD',TO_DATE('24-
SEP-2020','DD-MON-
YYYY'),'TODOROKI','SHOTO','4671524735151580',TO_DATE('AUG-2021','MON-
YYYY'),'826',6.56);
INSERT INTO PAYMENT VALUES('PY123456S','BK123456S','CARD',TO_DATE('23-
OCT-2020','DD-MON-
YYYY'),'IZUKU','MIDORIYA','4524075348975567',TO_DATE('AUG-2021','MON-
YYYY'),'699',38.29);
INSERT INTO PAYMENT VALUES('PY123456T','BK123456T','CASH',TO_DATE('23-
OCT-2020','DD-MON-YYYY'),'ICHIGO','KUROSAKI',NULL,NULL,NULL,5.27);

/* RECURRENT_BOOKING Inserts */
INSERT INTO RECURRENT_BOOKING VALUES('BK123456A','RB123456A');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456B','RB123456B');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456C','RB123456C');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456D','RB123456D');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456E','RB123456E');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456F','RB123456F');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456H','RB123456G');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456I','RB123456H');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456K','RB123456I');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456L','RB123456J');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456N','RB123456K');
INSERT INTO RECURRENT_BOOKING VALUES('BK1234560','RB123456L');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456P','RB123456M');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456R','RB123456N');
INSERT INTO RECURRENT_BOOKING VALUES('BK123456S','RB1234560');
```

Tables and code Used For Testing and Exponential Insets (Denormalised and Normalised)

```
/* DENORMALIZED TABLE */
DROP TABLE REGULAR_BOOKING_DNORM;
DROP TABLE CLIENTS_DNORM;
DROP TABLE CARS_DNORM;
DROP TABLE DRIVER_REVENUE_DNORM;
DROP TABLE BOOKING_DNORM;
DROP TABLE EMPLOYEE_INCIDENTS_DNORM;
DROP TABLE SHIFT_DNORM;
DROP TABLE EMPLOYEE_DNORM;

CREATE TABLE BOOKING_DNORM (
BOOKINGID CHAR(9) NOT NULL,
OPERATORID CHAR(9),
DRIVERID CHAR(9),
REGBOOKID CHAR(9),
CLIENTID CHAR(9),
DATEBOOKED DATE,
PICKUPDATE DATE,
PICKUPADDRESS VARCHAR(255),
DROPOFFADDRESS VARCHAR(255),
NOOFPASSENGERS NUMBER(2),
CUSTOMERPHONENO CHAR(11),
PAYMENTMETHOD VARCHAR(10),
PAYMENTDATE DATE,
PAYEESFIRSTNAME VARCHAR(255),
PAYEESLASTNAME VARCHAR(255),
CARDNO VARCHAR(16),
EXPIRYDATE DATE,
CVV CHAR(3),
AMOUNTPAID NUMBER(8,2)
);

CREATE TABLE SHIFT_DNORM (
```

```
SHIFTID CHAR(9) NOT NULL,
EMPLOYEEID CHAR(9),
EMPFNAME VARCHAR(255),
EMPLNAME VARCHAR(255),
EMPLOYEETYPE VARCHAR(20),
SHIFTDATE DATE,
SHIFTTYPE VARCHAR(20)
);
CREATE TABLE CARS_DNORM (
REGISTRATIONNO CHAR(7) NOT NULL,
DRIVERID CHAR(9),
REGISTRATIONYEAR INTEGER,
MILEAGE VARCHAR(10),
LATESTMOTDATE DATE,
NOOFSEATS VARCHAR(10),
CARSTATUS VARCHAR(30),
OWNERID VARCHAR(9)
);

CREATE TABLE DRIVER_REVENUE_DNORM (
REVENUEID CHAR(9) NOT NULL,
DRIVERID CHAR(9),
REVENUEDATE DATE,
GROSSTAKINGS NUMBER(8,2),
PAIDTOCOMPANY NUMBER(8,2),
NETEARNINGS NUMBER(8,2)
);

CREATE TABLE EMPLOYEE_DNORM (
NINUMBER CHAR(9) NOT NULL,
ADDRESS VARCHAR2(255),
POSTCODE VARCHAR(20),
CITY VARCHAR(50),
FIRSTNAME VARCHAR(255),
LASTNAME VARCHAR(255),
PPHONENO CHAR(11),
SPHONENO CHAR(11),
SEX CHAR(1),
DOB DATE,
EMAIL VARCHAR2(255),
HIREDATE DATE,
DRIVERLICENSEID CHAR(9),
PAYTYPE VARCHAR(9),
```

```
PAYVALUE NUMBER(9),
SALARY NUMBER(8,2),
TAXCODE VARCHAR(20)
);




CREATE TABLE REGULAR_BOOKING_DNORM (
REGBOOKID CHAR(9) NOT NULL,
CLIENTID CHAR(9),
BOOKINGFREQUENCY VARCHAR(7),
STARTDATE DATE,
DURATION VARCHAR(10),
NOOFPASSENGERS  NUMBER(2)
);
CREATE TABLE CLIENTS_DNORM (
CLIENTID CHAR(9) NOT NULL,
CLIENTTYPE VARCHAR(15),
CONTACTFNAME VARCHAR(255),
CONTACTLNAME VARCHAR(255),
PPHONENO CHAR(11),
SPHONENO CHAR(11),
CORPERATIONNAME VARCHAR(50),
EMAIL VARCHAR2(400)
);

CREATE TABLE EMPLOYEE_INCIDENTS_DNORM (
INCIDENTID CHAR(9) NOT NULL,
NINUMBER CHAR(9),
INCIDENTTYPE VARCHAR2(20),
INCIDENTNOTES VARCHAR2(4000),
INCIDENTDATE DATE
);

/* INSERTS */
/* INSERT INTO BOOKING_DNORM VALUES
('BK0000001','EP0000001','EP0000002', NULL,NULL,TO_DATE('08-DEC-
2019','DD-MON-YYYY'),TO_DATE('12-NOV-2020','DD-MON-YYYY'), 'PKUP 1
STREET EP CD1 LONDON','DRPOF 1 STREET EP CD1 LONDON',3,'07900000000',
'CASH', TO_DATE('11-NOV-2020','DD-MON-YYYY'), 'RAY', 'JAY', NULL, NULL,
NULL, 50.05); */
```

```
INSERT INTO BOOKING_DNORM VALUES ('BK0000002','EP0000003','EP0000004',
'RB0000001','CL123456A',TO_DATE('07-DEC-2019','DD-MON-
YYYY'),TO_DATE('11-NOV-2020','DD-MON-YYYY'), 'PKUP 2 STREET EP CD2
LONDON','DRPOF 2 STREET EP CD2 LONDON',5,'07900000001', 'CARD',
TO_DATE('09-NOV-2020','DD-MON-YYYY'), 'BEN', 'KEN', '4444555566667777',
TO_DATE('19-JAN-2025','DD-MON-YYYY'), '123', 40.66);
INSERT INTO BOOKING_DNORM VALUES ('BK0000003','EP0000005','EP0000006',
'RB0000003','CL123456B',TO_DATE('05-OCT-2018','DD-MON-
YYYY'),TO_DATE('11-MAR-2019','DD-MON-YYYY'), 'PKUP 3 STREET EP CD3
LONDON','DRPOF 3 STREET EP CD3 CARDIFF',5,'07900000002',
'CARD',TO_DATE('05-OCT-2018','DD-MON-YYYY'), 'DAN', 'TAN',
'1111555566667777', TO_DATE('19-JAN-2022','DD-MON-YYYY'), '130', 140);
INSERT INTO BOOKING_DNORM VALUES ('BK0000004','EP0000007','EP0000008',
'RB0000002','CL123456A',TO_DATE('08-AUG-2016','DD-MON-
YYYY'),TO_DATE('02-JAN-2020','DD-MON-YYYY'), 'PKUP 4 STREET EP CD4
LONDON','DRPOF 4 STREET EP CD4 LONDON',1,'07900000003', 'CASH',
TO_DATE('11-NOV-2020','DD-MON-YYYY'), 'AYA', 'YA', NULL, NULL, NULL,
6);

/* */

INSERT INTO REGULAR_BOOKING_DNORM VALUES
('RB000001','CL123456A','MONTHLY',TO_DATE('11-NOV-2020','DD-MON-
YYYY'),'30 DAYS','50');
INSERT INTO REGULAR_BOOKING_DNORM VALUES
('RB000003','CL123456B','WEEKLY',TO_DATE('11-JAN-2011','DD-MON-
YYYY'),'30 DAYS','2');
INSERT INTO REGULAR_BOOKING_DNORM VALUES
('RB000002','CL123456A','DAILY',TO_DATE('10-NOV-2020','DD-MON-
YYYY'),'14 DAYS','4');

/* */

INSERT INTO SHIFT_DNORM VALUES ('SH123456A', 'EP123456A','Dan',
'Fcface', 'OPERATOR',TO_DATE('02-NOV-2020', 'DD-MON-YYYY'), 'MORNING');
INSERT INTO SHIFT_DNORM VALUES ('SH123456B', 'EP123456B', 'Ray',
'Rampatino', 'OPERATOR',TO_DATE('04-NOV-2020', 'DD-MON-YYYY'),
'AFTERNOON');
INSERT INTO SHIFT_DNORM VALUES ('SH123456C', 'EP123456C','Digby',
'Dadarino', 'DRIVER',TO_DATE('08-NOV-2020', 'DD-MON-YYYY'), 'NIGHT');
INSERT INTO SHIFT_DNORM VALUES ('SH123456D', 'EP123456D','Dan2',
'Fcface2', 'OPERATOR',TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
'MORNING');
```

```
INSERT INTO SHIFT_DNORM VALUES ('SH123456E', 'EP123456E', 'Ray2',
'Rampatino2', 'OPERATOR',TO_DATE('04-NOV-2020', 'DD-MON-YYYY'),
'AFTERNOON');
INSERT INTO SHIFT_DNORM VALUES ('SH123456F', 'EP123456F','Digby2',
'Dadarino2', 'DRIVER',TO_DATE('08-NOV-2020', 'DD-MON-YYYY'),
'MORNING');

/* */

INSERT INTO CARS_DNORM
VALUES('CR1234A','DV123456A',2013,52041,TO_DATE('02-JAN-2020','DD-MON-
YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS_DNORM
VALUES('CR1234B','DV123456B',2019,43214,TO_DATE('08-JUL-2020','DD-MON-
YYYY'),7,'IN FOR SERVICE','DV123456B');
INSERT INTO CARS_DNORM
VALUES('CR1234C','DV123456C',2018,23791,TO_DATE('02-MAY-2020','DD-MON-
YYYY'),3,'ROADWORTHY','COMPANY*');

/* */

INSERT INTO DRIVER_REVENUE_DNORM VALUES
('RV123456A','DV123456A',TO_DATE('01-JUL-2020','DD-MON-
YYYY'),3000,476,2524);
INSERT INTO DRIVER_REVENUE_DNORM VALUES
('RV123456B','DV123456B',TO_DATE('22-AUG-2020','DD-MON-
YYYY'),2000,426,1574);
INSERT INTO DRIVER_REVENUE_DNORM VALUES
('RV123456C','DV123456C',TO_DATE('1-JUN-2020','DD-MON-
YYYY'),3000,393,2607);

/* */

INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456A','43 OCEANA CLOSE','N16
7GK','LONDON','FABIEN','GARNER','07000000028',NULL,'M',TO_DATE('05-AUG-
1974','DD-MON-YYYY'),'EMAIL1@EMAIL.COM',TO_DATE('08-JAN-2020','DD-MON-
YYYY'),'DV123456A','FIXEDFULL', 36000, NULL, NULL );
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456D','12 PARKLANE
AVENUE','N2F
0FL','LONDON','BRYAN','NAISMITH','07000000025',NULL,'M',TO_DATE('03-
MAY-1979','DD-MON-YYYY'),'EMAIL2@EMAIL.COM',TO_DATE('02-MAR-2020','DD-
MON-YYYY'),NULL,NULL,NULL, 40000, '1250L' );
```

```
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456B','1 BERKIN STREET','B1
1DA','BIRMINGHAM','AMY','HALLOW','07000000023',NULL,'F',TO_DATE('31-
MAY-1993','DD-MON-YYYY'),'EMAIL3@EMAIL.COM',TO_DATE('01-JUN-2020','DD-
MON-YYYY'),'DV123456B','FIXEDTEMP', 24000, NULL, NULL );
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456C','7 LADBROKE GROVE','W11
8FS','LONDON','GARY','JOHNSON','07000000022',NULL,'M',TO_DATE('25-JUL-
1992','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-
YYYY'),'DV123456C','FIXEDFULL', 36000, NULL, NULL );
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456D','7 LADBROKE GROVE','W11
8FS','LONDON','DARY','SON','07000000027',NULL,'M',TO_DATE('25-JUL-
1992','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-
YYYY'),'DV123456C','FIXEDFULL', 36000, NULL, NULL );
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456E','7 LADBROKE GROVE','W11
8FS','LONDON','RARY','ROHNSON','07000000026',NULL,'M',TO_DATE('25-JUL-
1992','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-
YYYY'),'DV123456C','FIXEDFULL', 36000, NULL, NULL );

/* */

INSERT INTO EMPLOYEE_INCIDENTS_DNORM VALUES
('IN123456A','EP123456A','SICK',NULL,TO_DATE('02-NOV-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_DNORM VALUES
('IN123456B','EP123456B','DISCIPLINARY ACTION','BROUGHT ILLEGAL
SUBSTANCES INTO THE WORKPLACE',TO_DATE('06-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_DNORM VALUES
('IN123456C','EP123456C','VACATION REQUEST','I HEAR SPAIN IS A GREAT
PLACE TO FIND CARS, MIND IF I TAKE A LOOK MYSELF?',TO_DATE('26-NOV-
2020','DD-MON-YYYY'));

/* */

INSERT INTO CLIENTS_DNORM VALUES
('CL123456A','PRIVATE','ALPHONSO','ROGERS','07500000013','07500000098',
NULL,'CLEMAIL1@EMAIL.COM');
INSERT INTO CLIENTS_DNORM VALUES
('CL123456B','CORPORATE','LIONEL','ALLAN','07500000012','07500000067','
PATEK INDUSTRIES','CLEMAIL2@EMAIL.COM');
INSERT INTO CLIENTS_DNORM VALUES
('CL123456C','PRIVATE','JAMIE','DAVIES','07500000011','07500000021',NUL
L,'CLEMAIL3@EMAIL.COM');
```

```
/* BIG TABLES */
DROP TABLE REGULAR_BOOKING_DNORM_BIG;
DROP TABLE CLIENTS_DNORM_BIG;
DROP TABLE CARS_DNORM_BIG;
DROP TABLE DRIVER_REVENUE_DNORM_BIG;
DROP TABLE BOOKING_DNORM_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_DNORM_BIG;
DROP TABLE SHIFT_DNORM_BIG;
DROP TABLE EMPLOYEE_DNORM_BIG;

CREATE TABLE REGULAR_BOOKING_DNORM_BIG AS SELECT * FROM
REGULAR_BOOKING_DNORM;
CREATE TABLE CLIENTS_DNORM_BIG AS SELECT * FROM CLIENTS_DNORM;
CREATE TABLE CARS_DNORM_BIG AS SELECT * FROM CARS_DNORM;
CREATE TABLE DRIVER_REVENUE_DNORM_BIG AS SELECT * FROM
DRIVER_REVENUE_DNORM;
CREATE TABLE BOOKING_DNORM_BIG AS SELECT * FROM BOOKING_DNORM;
CREATE TABLE EMPLOYEE_INCIDENTS_DNORM_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS_DNORM;
CREATE TABLE SHIFT_DNORM_BIG AS SELECT * FROM SHIFT_DNORM;
CREATE TABLE EMPLOYEE_DNORM_BIG AS SELECT * FROM EMPLOYEE_DNORM;

/* INCREASE EXPONENTIALLY */
SET SERVEROUTPUT ON;

DECLARE
cntr NUMBER := 0;
BEGIN
    FOR loop_counter IN 1..11 /*13*
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_DNORM_BIG SELECT * FROM BOOKING_DNORM_BIG;
```

```
INSERT INTO REGULAR_BOOKING_DNORM_BIG SELECT * FROM
REGULAR_BOOKING_DNORM_BIG;
INSERT INTO SHIFT_DNORM_BIG SELECT * FROM SHIFT_DNORM_BIG;
INSERT INTO CARS_DNORM_BIG SELECT * FROM CARS_DNORM_BIG;
INSERT INTO DRIVER_REVENUE_DNORM_BIG SELECT * FROM
DRIVER_REVENUE_DNORM_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_DNORM_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_DNORM_BIG;
INSERT INTO CLIENTS_DNORM_BIG SELECT * FROM CLIENTS_DNORM_BIG;
INSERT INTO EMPLOYEE_DNORM_BIG SELECT * FROM EMPLOYEE_DNORM_BIG;
END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/


/* TABLE COUNTS */
/* SELECT COUNT(*) FROM BOOKING_DNORM;
SELECT COUNT(*) FROM REGULAR_BOOKING_DNORM;
SELECT COUNT(*) FROM SHIFT_DNORM;
SELECT COUNT(*) FROM CARS_DNORM;
SELECT COUNT(*) FROM DRIVER_REVENUE_DNORM;
SELECT COUNT(*) FROM EMPLOYEE_INCIDENTS_DNORM;
SELECT COUNT(*) FROM CLIENTS_DNORM;

SELECT COUNT(*) FROM BOOKING_DNORM_BIG;
SELECT COUNT(*) FROM REGULAR_BOOKING_DNORM_BIG;
SELECT COUNT(*) FROM SHIFT_DNORM_BIG;
SELECT COUNT(*) FROM CARS_DNORM_BIG;
SELECT COUNT(*) FROM DRIVER_REVENUE_DNORM_BIG;
SELECT COUNT(*) FROM EMPLOYEE_INCIDENTS_DNORM_BIG;
SELECT COUNT(*) FROM CLIENTS_DNORM_BIG; */

/* NORMALIZED TABLE WITHOUT CONSTRAINTS */

/* DROP STATEMENTS */
DROP TABLE RECURRENT_BOOKING;
DROP TABLE PAYMENT;
DROP TABLE CORPORATE_CLIENT;
DROP TABLE REGULAR_BOOKING;
DROP TABLE CLIENTS;
DROP TABLE CARS;
DROP TABLE DRIVER_REVENUE;
```

```
DROP TABLE BOOKING;
DROP TABLE DRIVER;
DROP TABLE PAY_TYPE;
DROP TABLE OPERATOR;
DROP TABLE EMPLOYEE_INCIDENTS;
DROP TABLE SHIFT;
DROP TABLE EMPLOYEE;
DROP TABLE CITY_LOOKUP;



/* CITY_LOOKUP Table */
CREATE TABLE CITY_LOOKUP(
POSTCODE VARCHAR(8) PRIMARY KEY,
CITY VARCHAR(50)
);

/* CLIENTS Table */
CREATE TABLE CLIENTS (
CLIENTID CHAR(9) PRIMARY KEY,
CLIENTTYPE VARCHAR(15),
CONTACTFNAME VARCHAR(255),
CONTACTLNAME VARCHAR(255),
PPHONENO CHAR(11),
SPHONENO CHAR(11),
EMAIL VARCHAR2(400)
);

/* PAY_TYPE Table */
CREATE TABLE PAY_TYPE (
PAYTYPE VARCHAR(9) PRIMARY KEY,
PAYVALUE NUMBER(9)
);

/* EMPLOYEE Table */
CREATE TABLE EMPLOYEE(
NINUMBER CHAR(9) PRIMARY KEY,
POSTCODE VARCHAR(20),
ADDRESS VARCHAR2(255),
FIRSTNAME VARCHAR(255),
LASTNAME VARCHAR(255),
PPHONENO CHAR(11),
SPHONENO CHAR(11),
```

```
SEX CHAR(1),
DOB DATE,
EMAIL VARCHAR2(255),
HIREDATE DATE
);




/* OPERATOR Table */
CREATE TABLE OPERATOR (
OPERATORID CHAR(9),
EMPLOYEEID CHAR(9),
SALARY NUMBER(8,2),
TAXCODE VARCHAR(20)
);

/* DRIVER Table */
CREATE TABLE DRIVER (
DRIVERLICENSEID CHAR(9),
EMPLOYEEID CHAR(9),
PAYTYPE VARCHAR (9)
);

/* BOOKING Table */
CREATE TABLE BOOKING (
BOOKINGID CHAR(9) PRIMARY KEY,
OPERATORID CHAR(9),
DRIVERID CHAR(9),
DATEBOOKED DATE,
PICKUPDATE DATE,
PICKUPADDRESS VARCHAR(255),
DROPOFFADDRESS VARCHAR2(255),
NOOFPASSENGERS NUMBER(2),
CUSTOMERPHONENUMBER CHAR(11)
);

/* REGULAR_BOOKING Table */
CREATE TABLE REGULAR_BOOKING(
REGBOOKID CHAR(9) PRIMARY KEY,
CLIENTID CHAR(9),
```

```
BOOKINGFREQUENCY VARCHAR(50),
STARTDATE DATE,
NOOFTRANSPORTEES NUMBER(2),
DURATION VARCHAR(10)
);




/* DRIVER_REVENUE Table */
CREATE TABLE DRIVER_REVENUE(
REVENUEID CHAR(9) PRIMARY KEY,
DRIVERID CHAR(9),
REVENUEDATE DATE,
GROSSTAKINGS NUMBER(8,2),
PAIDTOCOMPANY NUMBER(8,2),
NETEARNINGS NUMBER(8,2)
);

/* CORPERATE_CLIENT Table */
CREATE TABLE CORPORATE_CLIENT (
CLIENTID CHAR(9) PRIMARY KEY,
CORPERATIONNAME VARCHAR(50)
);

/* EMPLOYEE_INCIDENTS Table */
CREATE TABLE EMPLOYEE_INCIDENTS (
INCIDENTID CHAR(9) PRIMARY KEY,
NINUMBER CHAR(9),
INCIDENTTYPE VARCHAR2(20),
INCIDENTNOTES VARCHAR2(4000),
INCIDENTDATE DATE
);

/* CARS Table */
CREATE TABLE CARS (
REGISTRATIONNO CHAR(7) PRIMARY KEY,
DRIVERID CHAR(9),
REGISTRATIONYEAR INTEGER,
MILEAGE VARCHAR(10),
LASTMOTDATE DATE,
```

```
NOOFSEATS VARCHAR(10),
CARSTATUS VARCHAR(30),
OWNERID VARCHAR(9)
);

/* SHIFT Table */
CREATE TABLE SHIFT (
SHIFTID CHAR(9) PRIMARY KEY,
EMPLOYEEID CHAR(9),
SHIFTDATE DATE,
SHIFTTYPE VARCHAR(20)
);

/* PAYMENT Table */
CREATE TABLE PAYMENT (
PAYMENTID CHAR(9) PRIMARY KEY,
BOOKINGID CHAR(9),
PAYMENTMETHOD VARCHAR(10),
PAYMENTDATE DATE,
PAYEESFIRSTNAME VARCHAR(255),
PAYEESLASTNAME VARCHAR(255),
CARDNUMBER VARCHAR(16),
EXPIRYDATE DATE,
CVV CHAR(3),
AMOUNTPAID NUMBER(8,2)
);

/* RECURRENT_BOOKING Table */
CREATE TABLE RECURRENT_BOOKING (
BOOKINGID CHAR(9),
REGBOOKID CHAR(9)
);

/* CITY_LOOKUP Inserts */
INSERT INTO CITY_LOOKUP VALUES ('N14 4AU', 'NORTH LONDON');
INSERT INTO CITY_LOOKUP VALUES ('NW8 9ZL', 'NORTH WEST LONDON');
INSERT INTO CITY_LOOKUP VALUES ('SW17 6AF', 'SOUTH WEST LONDON');

/* EMPLOYEE Inserts */
INSERT INTO EMPLOYEE VALUES ('EP123456A','B1 1DA','5 ALLINGTON
ROAD','Dan', 'Fcface','07000000000',NULL,'M',TO_DATE('23-JAN-2002','DD-
MON-YYYY'),'EMAIL1@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-YYYY'));
```

```
INSERT INTO EMPLOYEE VALUES ('EP123456B','B1 1DA','44 ALLINGTON ROAD',
'Ray', 'Rampatino','07000000001','07000000002','M',TO_DATE('12-JAN-
1998','DD-MON-YYYY'),'EMAIL2@EMAIL.COM',TO_DATE('18-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456C','B1 1DA','39 ALLINGTON
ROAD','Digby','Dadarino','07000000003',NULL,'M',TO_DATE('23-JAN-
2002','DD-MON-YYYY'),'EMAIL3@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456D','B1 1DA','60 ALLINGTON
ROAD','Dan2','Fcface2','07000000004',NULL,'M',TO_DATE('12-FEB-
2001','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456E','B1 1DA','90 ALLINGTON
ROAD','Ray2', 'Rampatino2','07000000005',NULL,'M',TO_DATE('12-JAN-
1971','DD-MON-YYYY'),'EMAIL5@EMAIL.COM',TO_DATE('01-JUN-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456F','N14 4AU','63 KEYSE
ROAD','Digby2',
'Dadarino2','07000000006','07000000007','M',TO_DATE('16-OCT-1978','DD-
MON-YYYY'),'EMAIL6@EMAIL.COM',TO_DATE('05-MAR-2020','DD-MON-YYYY'));

/* CLIENTS Inserts */
INSERT INTO CLIENTS VALUES
('CL123456A','PRIVATE','EDWARD','ELRIC','07500000000',NULL,'CLEMAIL1@EM
AIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456B','PRIVATE','HASHIRAMA','SENJU','07500000001',NULL,'CLEMAIL2
@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456C','PRIVATE','RINTAROU','OKABE','07500000002','07500000022','
CLEMAIL3@EMAIL.COM');

/* PAY_TYPE Inserts */
INSERT INTO PAY_TYPE VALUES ('FIXEDTEMP', 24000);
INSERT INTO PAY_TYPE VALUES ('FIXEDFULL', 36000);
INSERT INTO PAY_TYPE VALUES ('COMM75', 75);

/* OPERATOR Inserts */
INSERT INTO OPERATOR VALUES('OP123456A','EP123456A',40000,'1250L');
INSERT INTO OPERATOR VALUES('OP123456B','EP123456B',42500,'1250L');
INSERT INTO OPERATOR VALUES('OP123456C','EP123456C',39000,'1250L');

/* DRIVER Inserts */
```

```
INSERT INTO DRIVER VALUES ('DV123456A','EP123456D','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456B','EP123456E','COMM75');
INSERT INTO DRIVER VALUES ('DV123456C','EP123456F','FIXEDFULL');

/* BOOKING Inserts */
INSERT INTO BOOKING VALUES
('BK0000002','OP123456A','DV123456A',TO_DATE('08-NOV-2020','DD-MON-
YYYY'),TO_DATE('12-NOV-2020','DD-MON-YYYY'),'21 BALFE STREET','79
TEMPLE WAY',3,'07900000000');
INSERT INTO BOOKING VALUES
('BK0000003','OP123456B','DV123456B',TO_DATE('07-MAY-2020','DD-MON-
YYYY'),TO_DATE('08-MAY-2020','DD-MON-YYYY'),'78 TRAILLE STREET','76
BOAR LANE',5,'07900000001');
INSERT INTO BOOKING VALUES
('BK0000004','OP123456C','DV123456C',TO_DATE('15-APR-2020','DD-MON-
YYYY'),TO_DATE('15-APR-2020','DD-MON-YYYY'),'72 QUAY STREET','63 PRINCE
CONSORT ROAD',5,'07900000002');

/* REGULAR_BOOKING Inserts */
INSERT INTO REGULAR_BOOKING VALUES
('RB0000001','CL123456A','WEEKLY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),3,'7 WEEKS');
INSERT INTO REGULAR_BOOKING VALUES
('RB0000002','CL123456B','DAILY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),5,'4 DAYS');
INSERT INTO REGULAR_BOOKING VALUES
('RB0000003','CL123456A','WEEKLY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),5,'4 WEEKS');

/* DRIVER_REVENUE Inserts */
INSERT INTO DRIVER_REVENUE VALUES ('RV123456A','DV123456A',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,476,2524);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456B','DV123456B',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2932,833,2099);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456C','DV123456C',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,497,2503);

/* CORPERATE_CLIENT Inserts */
INSERT INTO CORPORATE_CLIENT VALUES('CL123456A','UMBRELLA CORP');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456B','WONKA INDUSTRIES');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456C','STARK INDUSTRIES');

/* EMPLOYEE_INCIDENTS Inserts */
```

```
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456A','EP123457A','SICK','GOT MY FINGERS STUCK IN A BOWLING
BALL',TO_DATE('02-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456B','EP123456B','VACATION REQUEST','I WANT TO GO ON MY
HONEYMOON',TO_DATE('10-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456C','EP123456C','DISCIPLINARY ACTION','CAME TO WORK DRUNK.
LICENSE SUSPENDED UNTIL FURTHER NOTICE',TO_DATE('02-OCT-2020','DD-MON-
YYYY'));
INSERT INTO CARS VALUES('CR1234A','DV123456A',2018,52041,TO_DATE('06-
AUG-2020','DD-MON-YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS VALUES('CR1234B','DV123456B',2008,112218,TO_DATE('28-
AUG-2020','DD-MON-YYYY'),4,'AWAITING REPAIR','COMPANY*');
INSERT INTO CARS VALUES('CR1234C','DV123456C',2012,178627,TO_DATE('28-
APR-2020','DD-MON-YYYY'),5,'ROADWORTHY','DV123456C');

/* SHIFT Inserts */
INSERT INTO SHIFT VALUES ('SH123456A','EP123456A',TO_DATE('05-NOV-
2020','DD-MON-YYYY'),'MORNING');
INSERT INTO SHIFT VALUES ('SH123456B','EP123456B',TO_DATE('25-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456C','EP123456C',TO_DATE('29-NOV-
2020','DD-MON-YYYY'),'NIGHT');
INSERT INTO SHIFT VALUES ('SH123456D','EP123456D',TO_DATE('05-NOV-
2020','DD-MON-YYYY'),'MORNING');
INSERT INTO SHIFT VALUES ('SH123456E','EP123456E',TO_DATE('13-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456F','EP123456F',TO_DATE('08-NOV-
2020','DD-MON-YYYY'),'MORNING');

/* PAYMENT Inserts */
INSERT INTO PAYMENT VALUES('PY123456A','BK0000002','CARD',TO_DATE('08-
NOV-2020','DD-MON-
YYYY'),'KAZUTO','KIRIGAYA','4432491216876558',TO_DATE('AUG-2023','MON-
YYYY'),'296',40.66);
INSERT INTO PAYMENT VALUES('PY123456B','BK0000003','CARD',TO_DATE('07-
MAY-2020','DD-MON-
YYYY'),'KURISU','MAKISE','5127071833675349',TO_DATE('AUG-2023','MON-
YYYY'),'314',140);
INSERT INTO PAYMENT VALUES('PY123456C','BK0000004','CARD',TO_DATE('15-
APR-2020','DD-MON-
```

```
YYYY'),'HACHIMAN','HIKIGAYA','5213211675421049',TO_DATE('AUG-
2023','MON-YYYY'),'329',6);

/* RECURRENT_BOOKING Inserts */
INSERT INTO RECURRENT_BOOKING VALUES('BK0000002','RB0000001');
INSERT INTO RECURRENT_BOOKING VALUES('BK0000003','RB0000002');
INSERT INTO RECURRENT_BOOKING VALUES('BK0000004','RB0000003');
/* BIG TABLES (Normalized) */
DROP TABLE REGULAR_BOOKING_BIG;
DROP TABLE CLIENTS_BIG;
DROP TABLE CARS_BIG;
DROP TABLE DRIVER_REVENUE_BIG;
DROP TABLE BOOKING_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_BIG;
DROP TABLE SHIFT_BIG;
DROP TABLE EMPLOYEE_BIG;

DROP TABLE RECURRENT_BOOKING_BIG;
DROP TABLE PAY_TYPE_BIG;
DROP TABLE DRIVER_BIG;
DROP TABLE OPERATOR_BIG;
DROP TABLE  CITY_LOOKUP_BIG;
DROP TABLE CORPORATE_CLIENT_BIG;
DROP TABLE PAYMENT_BIG;

CREATE TABLE REGULAR_BOOKING_BIG AS SELECT * FROM REGULAR_BOOKING;
CREATE TABLE CLIENTS_BIG AS SELECT * FROM CLIENTS;
CREATE TABLE CARS_BIG AS SELECT * FROM CARS;
CREATE TABLE DRIVER_REVENUE_BIG AS SELECT * FROM DRIVER_REVENUE;
CREATE TABLE BOOKING_BIG AS SELECT * FROM BOOKING;
CREATE TABLE EMPLOYEE_INCIDENTS_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS;
CREATE TABLE SHIFT_BIG AS SELECT * FROM SHIFT;
CREATE TABLE EMPLOYEE_BIG AS SELECT * FROM EMPLOYEE;

CREATE TABLE RECURRENT_BOOKING_BIG AS SELECT * FROM RECURRENT_BOOKING;
CREATE TABLE PAY_TYPE_BIG AS SELECT * FROM PAY_TYPE;
CREATE TABLE DRIVER_BIG AS SELECT * FROM DRIVER;
CREATE TABLE OPERATOR_BIG AS SELECT * FROM OPERATOR;
CREATE TABLE CITY_LOOKUP_BIG AS SELECT * FROM CITY_LOOKUP;
CREATE TABLE CORPORATE_CLIENT_BIG AS SELECT * FROM CORPORATE_CLIENT;
CREATE TABLE PAYMENT_BIG AS SELECT * FROM PAYMENT;
```

```
/* INCREASE EXPONENTIALLY (Normalized) */
SET SERVEROUTPUT ON;

DECLARE
cntr NUMBER := 0;
BEGIN
    FOR loop_counter IN 1..11 /*13*/
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_BIG SELECT * FROM BOOKING_BIG;
INSERT INTO REGULAR_BOOKING_BIG SELECT * FROM REGULAR_BOOKING_BIG;
INSERT INTO SHIFT_BIG SELECT * FROM SHIFT_BIG;
INSERT INTO CARS_BIG SELECT * FROM CARS_BIG;
INSERT INTO DRIVER_REVENUE_BIG SELECT * FROM DRIVER_REVENUE_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_BIG;
INSERT INTO CLIENTS_BIG SELECT * FROM CLIENTS_BIG;

INSERT INTO RECURRENT_BOOKING_BIG SELECT * FROM RECURRENT_BOOKING_BIG;
INSERT INTO PAY_TYPE_BIG SELECT * FROM PAY_TYPE_BIG ;
INSERT INTO DRIVER_BIG SELECT * FROM DRIVER_BIG;
INSERT INTO CITY_LOOKUP_BIG SELECT * FROM CITY_LOOKUP_BIG;
INSERT INTO CORPORATE_CLIENT_BIG SELECT * FROM CORPORATE_CLIENT_BIG;
INSERT INTO OPERATOR_BIG  SELECT * FROM OPERATOR_BIG;
INSERT INTO PAYMENT_BIG SELECT * FROM PAYMENT_BIG;
INSERT INTO EMPLOYEE_BIG SELECT * FROM EMPLOYEE_BIG;

END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/

INSERT INTO CITY_LOOKUP_BIG VALUES ('N14 4AU', 'NORTH LONDON');
INSERT INTO CITY_LOOKUP_BIG VALUES ('NW8 9ZL', 'NORTH WEST LONDON');
INSERT INTO CITY_LOOKUP_BIG VALUES ('SW17 6AF', 'SOUTH WEST LONDON');
INSERT INTO CITY_LOOKUP_BIG VALUES ('W10 4AD', 'WEST LONDON');
INSERT INTO CITY_LOOKUP_BIG VALUES ('EN5 7QZ', 'ENFIELD');
INSERT INTO CITY_LOOKUP_BIG VALUES ('L40 0AA', 'LIVERPOOL');
INSERT INTO CITY_LOOKUP_BIG VALUES ('M1 2ER', 'MANCHESTER');
INSERT INTO CITY_LOOKUP_BIG VALUES ('B1 1DA', 'BIRMINGHAM');
```

```
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456A','B1 1DA','5 ALLINGTON
ROAD','AADI','BACCI','07000000000',NULL,'M',TO_DATE('23-JAN-2002','DD-
MON-YYYY'),'EMAIL1@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456B','B1 1DA','44 ALLINGTON
ROAD','AALAM','BABEL','07000000001','07000000002','M',TO_DATE('12-JAN-
1998','DD-MON-YYYY'),'EMAIL2@EMAIL.COM',TO_DATE('18-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456C','B1 1DA','39 ALLINGTON
ROAD','ABBA','BACCARI','07000000003',NULL,'M',TO_DATE('23-JAN-
2002','DD-MON-YYYY'),'EMAIL3@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456D','B1 1DA','60 ALLINGTON
ROAD','ABDULLAH','BACCHUS','07000000004',NULL,'M',TO_DATE('12-FEB-
2001','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456E','B1 1DA','90 ALLINGTON
ROAD','AARON','BACHMAN','07000000005',NULL,'M',TO_DATE('12-JAN-
1971','DD-MON-YYYY'),'EMAIL5@EMAIL.COM',TO_DATE('01-JUN-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456F','N14 4AU','63 KEYSE
ROAD','BAKI','CADIZ','07000000006','07000000007','M',TO_DATE('16-OCT-
1978','DD-MON-YYYY'),'EMAIL6@EMAIL.COM',TO_DATE('05-MAR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456G','N14 4AU','23 KEYSE
ROAD','BALDWIN','CADLE','07000000008',NULL,'M',TO_DATE('11-FEB-
1986','DD-MON-YYYY'),'EMAIL7@EMAIL.COM',TO_DATE('04-AUG-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456H','N14 4AU','17 KEYSE
ROAD','BABETTE','CADOTTE','07000000009',NULL,'F',TO_DATE('04-OCT-
1993','DD-MON-YYYY'),'EMAIL8@EMAIL.COM',TO_DATE('04-AUG-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456I','N14 4AU','72 KEYSE
ROAD','BAHULA','CADORETTE','07000000010',NULL,'F',TO_DATE('25-SEP-
2002','DD-MON-YYYY'),'EMAIL9@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456J','N14 4AU','88 KEYSE
ROAD','BARBRA','CADOGEN','07000000011','07000000012','F',TO_DATE('21-
MAR-1990','DD-MON-YYYY'),'EMAIL10@EMAIL.COM',TO_DATE('04-JUN-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456K','NW8 9ZL','57 DUNSTAN
ROAD','CADEN','DERWIN','07000000013',NULL,'M',TO_DATE('21-NOV-
1972','DD-MON-YYYY'),'EMAIL11@EMAIL.COM',TO_DATE('16-JUN-2020','DD-MON-
YYYY'));
```

```
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456L','NW8 9ZL','75 DUNSTAN
ROAD','CALISSA','DESMOND','07000000014',NULL,'F',TO_DATE('29-OCT-
1981','DD-MON-YYYY'),'EMAIL12@EMAIL.COM',TO_DATE('16-MAR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456M','NW8 9ZL','12 DUNSTAN
ROAD','CAITLYN','DEXTER','07000000015','07000000018','F',TO_DATE('20-
AUG-2000','DD-MON-YYYY'),'EMAIL13@EMAIL.COM',TO_DATE('01-SEP-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456N','NW8 9ZL','93 DUNSTAN
ROAD','CAI','DILLON','07000000016',NULL,'M',TO_DATE('02-JAN-1975','DD-
MON-YYYY'),'EMAIL14@EMAIL.COM',TO_DATE('27-OCT-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456O','NW8 9ZL','73 DUNSTAN
ROAD','CAILYN','DIXIE','07000000017','07000000019','F',TO_DATE('17-DEC-
1973','DD-MON-YYYY'),'EMAIL15@EMAIL.COM',TO_DATE('06-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456P','M1 2ER','17 ADELAIDE
GROVE','DAGNA','EARWOOD','07000000018',NULL,'F',TO_DATE('19-DEC-
1972','DD-MON-YYYY'),'EMAIL16@EMAIL.COM',TO_DATE('30-OCT-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456Q','M1 2ER','12 ADELAIDE
GROVE','DAHLIA','EASTIN','07000000019',NULL,'F',TO_DATE('16-JAN-
1989','DD-MON-YYYY'),'EMAIL17@EMAIL.COM',TO_DATE('18-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456R','M1 2ER','14 ADELAIDE
GROVE','DALILA','EAKES','07000000020','07000000021','F',TO_DATE('11-
SEP-2002','DD-MON-YYYY'),'EMAIL18@EMAIL.COM',TO_DATE('07-NOV-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456S','M1 2ER','14 ADELAIDE
GROVE','DALLAS','EAKES','07000000021','07000000020','M',TO_DATE('09-
JUN-2002','DD-MON-YYYY'),'EMAIL19@EMAIL.COM',TO_DATE('07-NOV-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456T','M1 2ER','19 ADELAIDE
GROVE','DALIYA','EASTON','07000000022','07000000019','F',TO_DATE('2-
JAN-1977','DD-MON-YYYY'),'EMAIL20@EMAIL.COM',TO_DATE('07-NOV-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456U','SW17 6AF','32 ABBEY
MEWS','ED','FAHL','07000000023','07000000028','M',TO_DATE('28-NOV-
1985','DD-MON-YYYY'),'EMAIL21@EMAIL.COM',TO_DATE('21-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456V','SW17 6AF','48 ABBEY
MEWS','EDD','FADEL','07000000024',NULL,'M',TO_DATE('13-MAR-1975','DD-
MON-YYYY'),'EMAIL22@EMAIL.COM',TO_DATE('05-JUN-2020','DD-MON-YYYY'));
```

```
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456W','SW17 6AF','2 ABBEY
MEWS','EDDY','FLETCHER','07000000025',NULL,'M',TO_DATE('10-SEP-
1977','DD-MON-YYYY'),'EMAIL23@EMAIL.COM',TO_DATE('24-MAR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456X','SW17 6AF','50 ABBEY
MEWS','EDGAR','FITZ','07000000026',NULL,'M',TO_DATE('19-AUG-1980','DD-
MON-YYYY'),'EMAIL24@EMAIL.COM',TO_DATE('26-MAY-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123456Z','SW17 6AF','61 ABBEY
MEWS','EDWARD','FLOYD','07000000027',NULL,'M',TO_DATE('20-JAN-
1980','DD-MON-YYYY'),'EMAIL25@EMAIL.COM',TO_DATE('31-MAR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457A','L40 0AA','43 OCEANA
CLOSE','FABIEN','GARNER','07000000028',NULL,'M',TO_DATE('05-AUG-
1974','DD-MON-YYYY'),'EMAIL26@EMAIL.COM',TO_DATE('08-JAN-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457B','L40 0AA','93 OCEANA
CLOSE','FIDEL','GARFIELD','07000000029',NULL,'M',TO_DATE('16-SEP-
1981','DD-MON-YYYY'),'EMAIL27@EMAIL.COM',TO_DATE('26-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457C','L40 0AA','90 OCEANA
CLOSE','FARRAH','GARRICK','07000000030','07000000033','M',TO_DATE('23-
OCT-1992','DD-MON-YYYY'),'EMAIL28@EMAIL.COM',TO_DATE('03-JUL-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457D','L40 0AA','95 OCEANA
CLOSE','FAYIZ','GEORGE','07000000031',NULL,'M',TO_DATE('07-JUN-
1978','DD-MON-YYYY'),'EMAIL29@EMAIL.COM',TO_DATE('01-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457E','L40 0AA','25 OCEANA
CLOSE','FERLIN','GORAN','07000000034',NULL,'M',TO_DATE('06-APR-
2000','DD-MON-YYYY'),'EMAIL30@EMAIL.COM',TO_DATE('24-JUN-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457F','W10 4AD','6 PARRY
STREET','GABBY','HALSEY','07000000035',NULL,'F',TO_DATE('29-DEC-
1974','DD-MON-YYYY'),'EMAIL31@EMAIL.COM',TO_DATE('11-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457G','W10 4AD','1 PARRY
STREET','GILDA','HAMILTON','07000000036',NULL,'F',TO_DATE('18-AUG-
2000','DD-MON-YYYY'),'EMAIL32@EMAIL.COM',TO_DATE('22-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457H','W10 4AD','46 PARRY
STREET','GALILEO','HANSON','07000000037',NULL,'M',TO_DATE('23-MAR-
1986','DD-MON-YYYY'),'EMAIL33@EMAIL.COM',TO_DATE('22-APR-2020','DD-MON-
YYYY'));
```

```
INSERT INTO EMPLOYEE_BIG_BIG VALUES ('EP123457I','W10 4AD','81 PARRY
STREET','GARRY','HANSON','07000000038',NULL,'M',TO_DATE('18-NOV-
1976','DD-MON-YYYY'),'EMAIL34@EMAIL.COM',TO_DATE('23-APR-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457J','W10 4AD','28 PARRY
STREET','GARON','HAYES','07000000039','07000000040','M',TO_DATE('16-
DEC-1999','DD-MON-YYYY'),'EMAIL35@EMAIL.COM',TO_DATE('03-AUG-2020','DD-
MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457K','EN5 7QZ','16 BRISTOW
ROAD','GARON','INGRAM','07000000041',NULL,'M',TO_DATE('04-NOV-
1970','DD-MON-YYYY'),'EMAIL36@EMAIL.COM',TO_DATE('03-AUG-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457L','EN5 7QZ','96 BRISTOW
ROAD','GARON','IRVING','07000000042','07000000043','M',TO_DATE('24-JUL-
1988','DD-MON-YYYY'),'EMAIL37@EMAIL.COM',TO_DATE('03-AUG-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457M','EN5 7QZ','58 BRISTOW
ROAD','GARON','IBACH','07000000044','07000000045','M',TO_DATE('11-OCT-
1977','DD-MON-YYYY'),'EMAIL38@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457N','EN5 7QZ','86 BRISTOW
ROAD','GARON','IIDA','07000000046',NULL,'M',TO_DATE('22-OCT-1974','DD-
MON-YYYY'),'EMAIL39@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_BIG VALUES ('EP123457O','EN5 7QZ','48 BRISTOW
ROAD','GARON','IKEDA','07000000047',NULL,'M',TO_DATE('23-SEP-1991','DD-
MON-YYYY'),'EMAIL40@EMAIL.COM',TO_DATE('07-NOV-2020','DD-MON-YYYY'));

INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456A','EP123457I','SICK','GOT MY FINGERS STUCK IN A BOWLING
BALL',TO_DATE('02-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456B','EP123456M','VACATION REQUEST','I WANT TO GO ON MY
HONEYMOON',TO_DATE('10-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456C','EP123456G','DISCIPLINARY ACTION','CAME TO WORK DRUNK.
LICENSE SUSPENDED UNTIL FURTHER NOTICE',TO_DATE('02-OCT-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456D','EP123456X','SICK','MY GIRLFRIEND BIT ME IN A BAD
PLACE',TO_DATE('12-OCT-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456E','EP123457I','SICK','I HAVE A BLOCKED NOSE',TO_DATE('15-
MAY-2020','DD-MON-YYYY'));
```

```
INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456F','EP123457B','SICK','BITTEN BY A SNAKE',TO_DATE('28-APR-
2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456G','EP123457K','VACATION REQUEST','LONG OVERDUE FOR A BREAK,
WANT TO TAKE MY FAMILY TO HAWAII',TO_DATE('03-SEP-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456H','EP123456F','VACATION REQUEST','I HEAR SPAIN IS A GREAT
PLACE TO FIND CARS, MIND IF I TAKE A LOOK MYSELF?',TO_DATE('07-MAR-
2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456I','EP123457A','SICK','REALLY BAD HANGOVER',TO_DATE('09-APR-
2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_BIG VALUES
('IN123456J','EP123456Z','SICK','DRANK TOILET WATER, I DONT FEEL SO
GOOD',TO_DATE('03-NOV-2020','DD-MON-YYYY'));

SET TIMING ON
SET AUTOTRACE TRACEONLY

/* FIRST RUN */
SELECT EMPLOYEE_BIG.FIRSTNAME || ' ' || EMPLOYEE_BIG.LASTNAME AS
"Employee Name", EMPLOYEE_INCIDENTS_BIG.IncidentType,
EMPLOYEE_INCIDENTS_BIG.IncidentNotes,
EMPLOYEE_INCIDENTS_BIG.IncidentDate, CITY_LOOKUP_BIG.PostCode,
CITY_LOOKUP_BIG.City
FROM EMPLOYEE_INCIDENTS_BIG, CITY_LOOKUP_BIG, EMPLOYEE_BIG
WHERE EMPLOYEE_INCIDENTS_BIG.NINumber = EMPLOYEE_BIG.NINumber
AND CITY_LOOKUP_BIG.POSTCODE = EMPLOYEE_BIG.POSTCODE
AND (CITY_LOOKUP_BIG.PostCode LIKE 'N%' OR
CITY_LOOKUP_BIG.PostCode LIKE 'NW%' OR
CITY_LOOKUP_BIG.PostCode LIKE 'SW%' OR CITY_LOOKUP_BIG.PostCode LIKE
'W%');

/* SECOND RUN */
DROP INDEX EMPLOYEE_INDEX;
DROP INDEX CITY_LOOKUP_INDEX;
DROP INDEX EMPLOYEE_INCIDENTS_INDEX;

CREATE INDEX EMPLOYEE_INDEX ON EMPLOYEE_BIG (NINUMBER, POSTCODE);
CREATE INDEX CITY_LOOKUP_INDEX ON CITY_LOOKUP_BIG (POSTCODE);
```

```
CREATE INDEX EMPLOYEE_INCIDENTS_INDEX ON EMPLOYEE_INCIDENTS_BIG
(NINUMBER);

SELECT EMPLOYEE_BIG.FIRSTNAME || ' ' || EMPLOYEE_BIG.LASTNAME AS
"Employee Name", EMPLOYEE_INCIDENTS_BIG.IncidentType,
EMPLOYEE_INCIDENTS_BIG.IncidentNotes,
EMPLOYEE_INCIDENTS_BIG.IncidentDate, CITY_LOOKUP_BIG.PostCode,
CITY_LOOKUP_BIG.City
FROM EMPLOYEE_INCIDENTS_BIG, CITY_LOOKUP_BIG, EMPLOYEE_BIG
WHERE EMPLOYEE_INCIDENTS_BIG.NINumber = EMPLOYEE_BIG.NINumber
AND CITY_LOOKUP_BIG.POSTCODE = EMPLOYEE_BIG.POSTCODE
AND (CITY_LOOKUP_BIG.PostCode LIKE 'N%' OR
CITY_LOOKUP_BIG.PostCode LIKE 'NW%' OR
CITY_LOOKUP_BIG.PostCode LIKE 'SW%' OR CITY_LOOKUP_BIG.PostCode LIKE
'W%');
```

Normalisation and Denormalisation Tests

```
SET TIMING ON

/* DNORM 1 */
SELECT EmployeeID AS "Employee ID", EmpFName AS "First Name", EmpLName
AS "Last Name"
FROM SHIFT_DNORM_BIG
WHERE ShiftType IN ('NIGHT','AFTERNOON')
GROUP BY EmployeeID, EmpFName, EmpLName
ORDER BY EmployeeID;

/* NORM 1 */
SELECT SHIFT_BIG.EmployeeID AS "Employee ID", EMPLOYEE_BIG.FirstName AS
"First Name", EMPLOYEE_BIG.LastName AS "Last Name"
FROM SHIFT_BIG
INNER JOIN EMPLOYEE_BIG ON SHIFT_BIG.EmployeeID = EMPLOYEE_BIG.NINumber
WHERE SHIFT_BIG.ShiftType IN ('NIGHT','AFTERNOON')
GROUP BY SHIFT_BIG.EmployeeID, EMPLOYEE_BIG.FirstName,
EMPLOYEE_BIG.LastName
ORDER BY SHIFT_BIG.EmployeeID;

/* DNORM 2 */
SELECT ClientID AS "Client", COUNT(*) AS "Total bookings",
AVG(AmountPaid) AS "Average booking cost", MAX(AmountPaid) AS "Maximum
paid", MIN(AmountPaid) AS "Minimum paid"
FROM BOOKING_DNORM_BIG
WHERE ClientID IS NOT NULL
GROUP BY ClientID;

/* NORM 2.1 */
SELECT ClientID AS "Client", COUNT(*) AS "Total bookings",
AVG(AmountPaid) AS "Average booking cost", MAX(AmountPaid) AS "Maximum
paid", MIN(AmountPaid) AS "Minimum paid"
FROM BOOKING BK
```

```
INNER JOIN RECURRENT_BOOKING RECB ON BK.BookingID = RECB.BookingID
INNER JOIN REGULAR_BOOKING REGB ON RECB.RegBookID = REGB.RegBookID
INNER JOIN PAYMENT PY ON PY.BookingID = BK.BookingID
GROUP BY ClientID;




/* INCREASE SOMEWHAT EXPONENTIALLY (Normalized) */
DROP TABLE REGULAR_BOOKING_BIG;
DROP TABLE CLIENTS_BIG;
DROP TABLE CARS_BIG;
DROP TABLE DRIVER_REVENUE_BIG;
DROP TABLE BOOKING_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_BIG;
DROP TABLE SHIFT_BIG;
DROP TABLE EMPLOYEE_BIG;

DROP TABLE RECURRENT_BOOKING_BIG;
DROP TABLE PAY_TYPE_BIG;
DROP TABLE DRIVER_BIG;
DROP TABLE OPERATOR_BIG;
DROP TABLE  CITY_LOOKUP_BIG;
DROP TABLE CORPORATE_CLIENT_BIG;
DROP TABLE PAYMENT_BIG;

CREATE TABLE REGULAR_BOOKING_BIG AS SELECT * FROM REGULAR_BOOKING;
CREATE TABLE CLIENTS_BIG AS SELECT * FROM CLIENTS;
CREATE TABLE CARS_BIG AS SELECT * FROM CARS;
CREATE TABLE DRIVER_REVENUE_BIG AS SELECT * FROM DRIVER_REVENUE;
CREATE TABLE BOOKING_BIG AS SELECT * FROM BOOKING;
CREATE TABLE EMPLOYEE_INCIDENTS_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS;
CREATE TABLE SHIFT_BIG AS SELECT * FROM SHIFT;
CREATE TABLE EMPLOYEE_BIG AS SELECT * FROM EMPLOYEE;

CREATE TABLE RECURRENT_BOOKING_BIG AS SELECT * FROM RECURRENT_BOOKING;
CREATE TABLE PAY_TYPE_BIG AS SELECT * FROM PAY_TYPE;
CREATE TABLE DRIVER_BIG AS SELECT * FROM DRIVER;
CREATE TABLE OPERATOR_BIG AS SELECT * FROM OPERATOR;
CREATE TABLE CITY_LOOKUP_BIG AS SELECT * FROM CITY_LOOKUP;
CREATE TABLE CORPORATE_CLIENT_BIG AS SELECT * FROM CORPORATE_CLIENT;
CREATE TABLE PAYMENT_BIG AS SELECT * FROM PAYMENT;
```

```
SET SERVEROUTPUT ON;

DECLARE
cntr NUMBER := 0;
BEGIN
    FOR loop_counter IN 1..5
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_BIG SELECT * FROM BOOKING_BIG;
INSERT INTO REGULAR_BOOKING_BIG SELECT * FROM REGULAR_BOOKING_BIG;
INSERT INTO SHIFT_BIG SELECT * FROM SHIFT_BIG;
INSERT INTO CARS_BIG SELECT * FROM CARS_BIG;
INSERT INTO DRIVER_REVENUE_BIG SELECT * FROM DRIVER_REVENUE_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_BIG;
INSERT INTO CLIENTS_BIG SELECT * FROM CLIENTS_BIG;

INSERT INTO RECURRENT_BOOKING_BIG SELECT * FROM RECURRENT_BOOKING_BIG;
INSERT INTO PAY_TYPE_BIG SELECT * FROM PAY_TYPE_BIG ;
INSERT INTO DRIVER_BIG SELECT * FROM DRIVER_BIG;
INSERT INTO CITY_LOOKUP_BIG SELECT * FROM CITY_LOOKUP_BIG;
INSERT INTO CORPORATE_CLIENT_BIG SELECT * FROM CORPORATE_CLIENT_BIG;
INSERT INTO OPERATOR_BIG  SELECT * FROM OPERATOR_BIG;
INSERT INTO PAYMENT_BIG SELECT * FROM PAYMENT_BIG;

END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/

/* NORM 2.2 */
SELECT REGB.ClientID AS “Client”, COUNT(*) AS “Total bookings”,
AVG(PY.AmountPaid) AS “Average booking cost”, MAX(PY.AmountPaid) AS
“Maximum paid”, MIN(PY.AmountPaid) AS “Minimum paid”
FROM BOOKING_BIG BK
```

```
INNER JOIN RECURRENT_BOOKING_BIG RECB ON BK.BookingID = RECB.BookingID
INNER JOIN REGULAR_BOOKING_BIG REGB ON RECB.RegBookID = REGB.RegBookID
INNER JOIN PAYMENT_BIG PY ON PY.BookingID = BK.BookingID
GROUP BY ClientID;

/* DNORM 3 */
UPDATE EMPLOYEE_DNORM_BIG
SET
    NINUMBER = ROWNUM,
ADDRESS = '123 STREET',
POSTCODE = ROWNUM,
CITY = 'LONDON',
FIRSTNAME = 'NAME1',
LASTNAME = 'NAME2',
PPHONENO = '01234567891',
SPHONENO = '01234567891',
SEX = 'F',
DOB = TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
EMAIL = 'EXAMPLE@EMAIL.COM',
HIREDATE  = TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
DRIVERLICENSEID = NULL,
PAYTYPE = NULL,
PAYVALUE = NULL,
SALARY = NULL,
TAXCODE = NULL;

/* NORM 3 */
UPDATE EMPLOYEE_BIG
SET
NINUMBER = ROWNUM,
ADDRESS = '123 STREET',
POSTCODE = ROWNUM,
FIRSTNAME = 'NAME1',
LASTNAME = 'NAME2',
PPHONENO = '01234567891',
SPHONENO = '01234567891',
SEX = 'F',
DOB = TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
EMAIL = 'EXAMPLE@EMAIL.COM',
HIREDATE  = TO_DATE('02-NOV-2020', 'DD-MON-YYYY');
```

```
/* INCREASE SOMEWHAT EXPONENTIALLY (Normalized and denormalized) */

DROP TABLE REGULAR_BOOKING_BIG;
DROP TABLE CLIENTS_BIG;
DROP TABLE CARS_BIG;
DROP TABLE DRIVER_REVENUE_BIG;
DROP TABLE BOOKING_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_BIG;
DROP TABLE SHIFT_BIG;
DROP TABLE EMPLOYEE_BIG;

DROP TABLE RECURRENT_BOOKING_BIG;
DROP TABLE PAY_TYPE_BIG;
DROP TABLE DRIVER_BIG;
DROP TABLE OPERATOR_BIG;
DROP TABLE  CITY_LOOKUP_BIG;
DROP TABLE CORPORATE_CLIENT_BIG;
DROP TABLE PAYMENT_BIG;

CREATE TABLE REGULAR_BOOKING_BIG AS SELECT * FROM REGULAR_BOOKING;
CREATE TABLE CLIENTS_BIG AS SELECT * FROM CLIENTS;
CREATE TABLE CARS_BIG AS SELECT * FROM CARS;
CREATE TABLE DRIVER_REVENUE_BIG AS SELECT * FROM DRIVER_REVENUE;
CREATE TABLE BOOKING_BIG AS SELECT * FROM BOOKING;
CREATE TABLE EMPLOYEE_INCIDENTS_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS;
CREATE TABLE SHIFT_BIG AS SELECT * FROM SHIFT;
CREATE TABLE EMPLOYEE_BIG AS SELECT * FROM EMPLOYEE;

CREATE TABLE RECURRENT_BOOKING_BIG AS SELECT * FROM RECURRENT_BOOKING;
CREATE TABLE PAY_TYPE_BIG AS SELECT * FROM PAY_TYPE;
CREATE TABLE DRIVER_BIG AS SELECT * FROM DRIVER;
CREATE TABLE OPERATOR_BIG AS SELECT * FROM OPERATOR;
CREATE TABLE CITY_LOOKUP_BIG AS SELECT * FROM CITY_LOOKUP;
CREATE TABLE CORPORATE_CLIENT_BIG AS SELECT * FROM CORPORATE_CLIENT;
```

```
CREATE TABLE PAYMENT_BIG AS SELECT * FROM PAYMENT;

SET SERVEROUTPUT ON;

DECLARE
cntr NUMBER := 0;
BEGIN
    FOR loop_counter IN 1..5
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_BIG SELECT * FROM BOOKING_BIG;
INSERT INTO REGULAR_BOOKING_BIG SELECT * FROM REGULAR_BOOKING_BIG;
INSERT INTO SHIFT_BIG SELECT * FROM SHIFT_BIG;
INSERT INTO CARS_BIG SELECT * FROM CARS_BIG;
INSERT INTO DRIVER_REVENUE_BIG SELECT * FROM DRIVER_REVENUE_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_BIG;
INSERT INTO CLIENTS_BIG SELECT * FROM CLIENTS_BIG;

INSERT INTO RECURRENT_BOOKING_BIG SELECT * FROM RECURRENT_BOOKING_BIG;
INSERT INTO PAY_TYPE_BIG SELECT * FROM PAY_TYPE_BIG ;
INSERT INTO DRIVER_BIG SELECT * FROM DRIVER_BIG;
INSERT INTO CITY_LOOKUP_BIG SELECT * FROM CITY_LOOKUP_BIG;
INSERT INTO CORPORATE_CLIENT_BIG SELECT * FROM CORPORATE_CLIENT_BIG;
INSERT INTO OPERATOR_BIG  SELECT * FROM OPERATOR_BIG;
INSERT INTO PAYMENT_BIG SELECT * FROM PAYMENT_BIG;

END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/

DROP TABLE REGULAR_BOOKING_DNORM_BIG;
DROP TABLE CLIENTS_DNORM_BIG;
DROP TABLE CARS_DNORM_BIG;
DROP TABLE DRIVER_REVENUE_DNORM_BIG;
DROP TABLE BOOKING_DNORM_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_DNORM_BIG;
DROP TABLE SHIFT_DNORM_BIG;
DROP TABLE EMPLOYEE_DNORM_BIG;

CREATE TABLE REGULAR_BOOKING_DNORM_BIG AS SELECT * FROM
REGULAR_BOOKING_DNORM;
```

```
CREATE TABLE CLIENTS_DNORM_BIG AS SELECT * FROM CLIENTS_DNORM;
CREATE TABLE CARS_DNORM_BIG AS SELECT * FROM CARS_DNORM;
CREATE TABLE DRIVER_REVENUE_DNORM_BIG AS SELECT * FROM
DRIVER_REVENUE_DNORM;
CREATE TABLE BOOKING_DNORM_BIG AS SELECT * FROM BOOKING_DNORM;
CREATE TABLE EMPLOYEE_INCIDENTS_DNORM_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS_DNORM;
CREATE TABLE SHIFT_DNORM_BIG AS SELECT * FROM SHIFT_DNORM;
CREATE TABLE EMPLOYEE_DNORM_BIG AS SELECT * FROM EMPLOYEE_DNORM;

SET SERVEROUTPUT ON;

DECLARE
cntr NUMBER := 0;
BEGIN
    FOR loop_counter IN 1..7
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_DNORM_BIG SELECT * FROM BOOKING_DNORM_BIG;
INSERT INTO REGULAR_BOOKING_DNORM_BIG SELECT * FROM
REGULAR_BOOKING_DNORM_BIG;
INSERT INTO SHIFT_DNORM_BIG SELECT * FROM SHIFT_DNORM_BIG;
INSERT INTO CARS_DNORM_BIG SELECT * FROM CARS_DNORM_BIG;
INSERT INTO DRIVER_REVENUE_DNORM_BIG SELECT * FROM
DRIVER_REVENUE_DNORM_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_DNORM_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_DNORM_BIG;
INSERT INTO CLIENTS_DNORM_BIG SELECT * FROM CLIENTS_DNORM_BIG;
END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/

SET TIMING ON
```

```sql
/* DNORM 1 */
SELECT EmployeeID AS "Employee ID", EmpFName AS "First Name", EmpLName
AS "Last Name"
FROM SHIFT_DNORM_BIG
WHERE ShiftType IN ('NIGHT','AFTERNOON')
GROUP BY EmployeeID, EmpFName, EmpLName
ORDER BY EmployeeID;

/* NORM 1 */
SELECT SHIFT_BIG.EmployeeID AS "Employee ID", EMPLOYEE_BIG.FirstName AS
"First Name", EMPLOYEE_BIG.LastName AS "Last Name"
FROM SHIFT_BIG
INNER JOIN EMPLOYEE_BIG ON SHIFT_BIG.EmployeeID = EMPLOYEE_BIG.NINumber
WHERE SHIFT_BIG.ShiftType IN ('NIGHT','AFTERNOON')
GROUP BY SHIFT_BIG.EmployeeID, EMPLOYEE_BIG.FirstName,
EMPLOYEE_BIG.LastName
ORDER BY SHIFT_BIG.EmployeeID;

/* DNORM 2 */
SELECT ClientID AS "Client", COUNT(*) AS "Total bookings",
AVG(AmountPaid) AS "Average booking cost", MAX(AmountPaid) AS "Maximum
paid", MIN(AmountPaid) AS "Minimum paid"
FROM BOOKING_DNORM_BIG
WHERE ClientID IS NOT NULL
GROUP BY ClientID;

/* NORM 2.1 */
SELECT ClientID AS "Client", COUNT(*) AS "Total bookings",
AVG(AmountPaid) AS "Average booking cost", MAX(AmountPaid) AS "Maximum
paid", MIN(AmountPaid) AS "Minimum paid"
FROM BOOKING BK
INNER JOIN RECURRENT_BOOKING RECB ON BK.BookingID = RECB.BookingID
INNER JOIN REGULAR_BOOKING REGB ON RECB.RegBookID = REGB.RegBookID
INNER JOIN PAYMENT PY ON PY.BookingID = BK.BookingID
GROUP BY ClientID;
```

```
/* INCREASE SOMEWHAT EXPONENTIALLY (Normalized) */
DROP TABLE REGULAR_BOOKING_BIG;
DROP TABLE CLIENTS_BIG;
DROP TABLE CARS_BIG;
DROP TABLE DRIVER_REVENUE_BIG;
DROP TABLE BOOKING_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_BIG;
DROP TABLE SHIFT_BIG;
DROP TABLE EMPLOYEE_BIG;

DROP TABLE RECURRENT_BOOKING_BIG;
DROP TABLE PAY_TYPE_BIG;
DROP TABLE DRIVER_BIG;
DROP TABLE OPERATOR_BIG;
DROP TABLE  CITY_LOOKUP_BIG;
DROP TABLE CORPORATE_CLIENT_BIG;
DROP TABLE PAYMENT_BIG;

CREATE TABLE REGULAR_BOOKING_BIG AS SELECT * FROM REGULAR_BOOKING;
CREATE TABLE CLIENTS_BIG AS SELECT * FROM CLIENTS;
CREATE TABLE CARS_BIG AS SELECT * FROM CARS;
CREATE TABLE DRIVER_REVENUE_BIG AS SELECT * FROM DRIVER_REVENUE;
CREATE TABLE BOOKING_BIG AS SELECT * FROM BOOKING;
CREATE TABLE EMPLOYEE_INCIDENTS_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS;
CREATE TABLE SHIFT_BIG AS SELECT * FROM SHIFT;
CREATE TABLE EMPLOYEE_BIG AS SELECT * FROM EMPLOYEE;

CREATE TABLE RECURRENT_BOOKING_BIG AS SELECT * FROM RECURRENT_BOOKING;
CREATE TABLE PAY_TYPE_BIG AS SELECT * FROM PAY_TYPE;
CREATE TABLE DRIVER_BIG AS SELECT * FROM DRIVER;
CREATE TABLE OPERATOR_BIG AS SELECT * FROM OPERATOR;
CREATE TABLE CITY_LOOKUP_BIG AS SELECT * FROM CITY_LOOKUP;
CREATE TABLE CORPORATE_CLIENT_BIG AS SELECT * FROM CORPORATE_CLIENT;
CREATE TABLE PAYMENT_BIG AS SELECT * FROM PAYMENT;
```

```
SET SERVEROUTPUT ON;




DECLARE
cntr NUMBER := 0;
BEGIN
    FOR loop_counter IN 1..5
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_BIG SELECT * FROM BOOKING_BIG;
INSERT INTO REGULAR_BOOKING_BIG SELECT * FROM REGULAR_BOOKING_BIG;
INSERT INTO SHIFT_BIG SELECT * FROM SHIFT_BIG;
INSERT INTO CARS_BIG SELECT * FROM CARS_BIG;
INSERT INTO DRIVER_REVENUE_BIG SELECT * FROM DRIVER_REVENUE_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_BIG;
INSERT INTO CLIENTS_BIG SELECT * FROM CLIENTS_BIG;

INSERT INTO RECURRENT_BOOKING_BIG SELECT * FROM RECURRENT_BOOKING_BIG;
INSERT INTO PAY_TYPE_BIG SELECT * FROM PAY_TYPE_BIG ;
INSERT INTO DRIVER_BIG SELECT * FROM DRIVER_BIG;
INSERT INTO CITY_LOOKUP_BIG SELECT * FROM CITY_LOOKUP_BIG;
INSERT INTO CORPORATE_CLIENT_BIG SELECT * FROM CORPORATE_CLIENT_BIG;
INSERT INTO OPERATOR_BIG  SELECT * FROM OPERATOR_BIG;
INSERT INTO PAYMENT_BIG SELECT * FROM PAYMENT_BIG;

END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/

/* NORM 2.2 */
SELECT REGB.ClientID AS “Client”, COUNT(*) AS “Total bookings”,
AVG(PY.AmountPaid) AS “Average booking cost”, MAX(PY.AmountPaid) AS
“Maximum paid”, MIN(PY.AmountPaid) AS “Minimum paid”
FROM BOOKING_BIG BK
INNER JOIN RECURRENT_BOOKING_BIG RECB ON BK.BookingID = RECB.BookingID
INNER JOIN REGULAR_BOOKING_BIG REGB ON RECB.RegBookID = REGB.RegBookID
```

```
INNER JOIN PAYMENT_BIG PY ON PY.BookingID = BK.BookingID
GROUP BY ClientID;









/* DNORM 3 */
UPDATE EMPLOYEE_DNORM_BIG
SET
    NINUMBER = ROWNUM,
ADDRESS = '123 STREET',
POSTCODE = ROWNUM,
CITY = 'LONDON',
FIRSTNAME = 'NAME1',
LASTNAME = 'NAME2',
PPHONENO = '01234567891',
SPHONENO = '01234567891',
SEX = 'F',
DOB = TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
EMAIL = 'EXAMPLE@EMAIL.COM',
HIREDATE  = TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
DRIVERLICENSEID = NULL,
PAYTYPE = NULL,
PAYVALUE = NULL,
SALARY = NULL,
TAXCODE = NULL;

/* NORM 3 */
UPDATE EMPLOYEE_BIG
SET
NINUMBER = ROWNUM,
ADDRESS = '123 STREET',
POSTCODE = ROWNUM,
FIRSTNAME = 'NAME1',
LASTNAME = 'NAME2',
PPHONENO = '01234567891',
SPHONENO = '01234567891',
SEX = 'F',
DOB = TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
EMAIL = 'EXAMPLE@EMAIL.COM',
HIREDATE  = TO_DATE('02-NOV-2020', 'DD-MON-YYYY');
```

```
/* INCREASE SOMEWHAT EXPONENTIALLY (Normalized and denormalized) */

DROP TABLE REGULAR_BOOKING_BIG;
DROP TABLE CLIENTS_BIG;
DROP TABLE CARS_BIG;
DROP TABLE DRIVER_REVENUE_BIG;
DROP TABLE BOOKING_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_BIG;
DROP TABLE SHIFT_BIG;
DROP TABLE EMPLOYEE_BIG;

DROP TABLE RECURRENT_BOOKING_BIG;
DROP TABLE PAY_TYPE_BIG;
DROP TABLE DRIVER_BIG;
DROP TABLE OPERATOR_BIG;
DROP TABLE  CITY_LOOKUP_BIG;
DROP TABLE CORPORATE_CLIENT_BIG;
DROP TABLE PAYMENT_BIG;

CREATE TABLE REGULAR_BOOKING_BIG AS SELECT * FROM REGULAR_BOOKING;
CREATE TABLE CLIENTS_BIG AS SELECT * FROM CLIENTS;
CREATE TABLE CARS_BIG AS SELECT * FROM CARS;
CREATE TABLE DRIVER_REVENUE_BIG AS SELECT * FROM DRIVER_REVENUE;
CREATE TABLE BOOKING_BIG AS SELECT * FROM BOOKING;
CREATE TABLE EMPLOYEE_INCIDENTS_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS;
CREATE TABLE SHIFT_BIG AS SELECT * FROM SHIFT;
CREATE TABLE EMPLOYEE_BIG AS SELECT * FROM EMPLOYEE;

CREATE TABLE RECURRENT_BOOKING_BIG AS SELECT * FROM RECURRENT_BOOKING;
CREATE TABLE PAY_TYPE_BIG AS SELECT * FROM PAY_TYPE;
CREATE TABLE DRIVER_BIG AS SELECT * FROM DRIVER;
CREATE TABLE OPERATOR_BIG AS SELECT * FROM OPERATOR;
CREATE TABLE CITY_LOOKUP_BIG AS SELECT * FROM CITY_LOOKUP;
CREATE TABLE CORPORATE_CLIENT_BIG AS SELECT * FROM CORPORATE_CLIENT;
CREATE TABLE PAYMENT_BIG AS SELECT * FROM PAYMENT;

SET SERVEROUTPUT ON;

DECLARE
cntr NUMBER := 0;
```

```
BEGIN
    FOR loop_counter IN 1..5
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_BIG SELECT * FROM BOOKING_BIG;
INSERT INTO REGULAR_BOOKING_BIG SELECT * FROM REGULAR_BOOKING_BIG;
INSERT INTO SHIFT_BIG SELECT * FROM SHIFT_BIG;
INSERT INTO CARS_BIG SELECT * FROM CARS_BIG;
INSERT INTO DRIVER_REVENUE_BIG SELECT * FROM DRIVER_REVENUE_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_BIG;
INSERT INTO CLIENTS_BIG SELECT * FROM CLIENTS_BIG;

INSERT INTO RECURRENT_BOOKING_BIG SELECT * FROM RECURRENT_BOOKING_BIG;
INSERT INTO PAY_TYPE_BIG SELECT * FROM PAY_TYPE_BIG ;
INSERT INTO DRIVER_BIG SELECT * FROM DRIVER_BIG;
INSERT INTO CITY_LOOKUP_BIG SELECT * FROM CITY_LOOKUP_BIG;
INSERT INTO CORPORATE_CLIENT_BIG SELECT * FROM CORPORATE_CLIENT_BIG;
INSERT INTO OPERATOR_BIG  SELECT * FROM OPERATOR_BIG;
INSERT INTO PAYMENT_BIG SELECT * FROM PAYMENT_BIG;

END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/

DROP TABLE REGULAR_BOOKING_DNORM_BIG;
DROP TABLE CLIENTS_DNORM_BIG;
DROP TABLE CARS_DNORM_BIG;
DROP TABLE DRIVER_REVENUE_DNORM_BIG;
DROP TABLE BOOKING_DNORM_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_DNORM_BIG;
DROP TABLE SHIFT_DNORM_BIG;
DROP TABLE EMPLOYEE_DNORM_BIG;

CREATE TABLE REGULAR_BOOKING_DNORM_BIG AS SELECT * FROM
REGULAR_BOOKING_DNORM;
CREATE TABLE CLIENTS_DNORM_BIG AS SELECT * FROM CLIENTS_DNORM;
CREATE TABLE CARS_DNORM_BIG AS SELECT * FROM CARS_DNORM;
CREATE TABLE DRIVER_REVENUE_DNORM_BIG AS SELECT * FROM
DRIVER_REVENUE_DNORM;
CREATE TABLE BOOKING_DNORM_BIG AS SELECT * FROM BOOKING_DNORM;
```

```
CREATE TABLE EMPLOYEE_INCIDENTS_DNORM_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS_DNORM;
CREATE TABLE SHIFT_DNORM_BIG AS SELECT * FROM SHIFT_DNORM;
CREATE TABLE EMPLOYEE_DNORM_BIG AS SELECT * FROM EMPLOYEE_DNORM;




SET SERVEROUTPUT ON;

DECLARE
cntr NUMBER := 0;
BEGIN
    FOR loop_counter IN 1..7
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_DNORM_BIG SELECT * FROM BOOKING_DNORM_BIG;
INSERT INTO REGULAR_BOOKING_DNORM_BIG SELECT * FROM
REGULAR_BOOKING_DNORM_BIG;
INSERT INTO SHIFT_DNORM_BIG SELECT * FROM SHIFT_DNORM_BIG;
INSERT INTO CARS_DNORM_BIG SELECT * FROM CARS_DNORM_BIG;
INSERT INTO DRIVER_REVENUE_DNORM_BIG SELECT * FROM
DRIVER_REVENUE_DNORM_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_DNORM_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_DNORM_BIG;
INSERT INTO CLIENTS_DNORM_BIG SELECT * FROM CLIENTS_DNORM_BIG;
END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/
```

Quick Test Code (Code used to quickly test normalised and denormalized tables)

```
/* DENORMALIZED TABLE */
DROP TABLE REGULAR_BOOKING_DNORM;
DROP TABLE CLIENTS_DNORM;
DROP TABLE CARS_DNORM;
DROP TABLE DRIVER_REVENUE_DNORM;
DROP TABLE BOOKING_DNORM;
DROP TABLE EMPLOYEE_INCIDENTS_DNORM;
DROP TABLE SHIFT_DNORM;
DROP TABLE EMPLOYEE_DNORM;

CREATE TABLE BOOKING_DNORM (
BOOKINGID CHAR(9) NOT NULL,
OPERATORID CHAR(9),
DRIVERID CHAR(9),
REGBOOKID CHAR(9),
CLIENTID CHAR(9),
DATEBOOKED DATE,
PICKUPDATE DATE,
PICKUPADDRESS VARCHAR(255),
DROPOFFADDRESS VARCHAR(255),
NOOFPASSENGERS NUMBER(2),
CUSTOMERPHONENO CHAR(11),
PAYMENTMETHOD VARCHAR(10),
PAYMENTDATE DATE,
PAYEESFIRSTNAME VARCHAR(255),
PAYEESLASTNAME VARCHAR(255),
CARDNO VARCHAR(16),
EXPIRYDATE DATE,
CVV CHAR(3),
AMOUNTPAID NUMBER(8,2)
);
```

```
CREATE TABLE SHIFT_DNORM (
SHIFTID CHAR(9) NOT NULL,
EMPLOYEEID CHAR(9),
EMPFNAME VARCHAR(255),
EMPLNAME VARCHAR(255),
EMPLOYEETYPE VARCHAR(20),
SHIFTDATE DATE,
SHIFTTYPE VARCHAR(20)
);
CREATE TABLE CARS_DNORM (
REGISTRATIONNO CHAR(7) NOT NULL,
DRIVERID CHAR(9),
REGISTRATIONYEAR INTEGER,
MILEAGE VARCHAR(10),
LATESTMOTDATE DATE,
NOOFSEATS VARCHAR(10),
CARSTATUS VARCHAR(30),
OWNERID VARCHAR(9)
);

CREATE TABLE DRIVER_REVENUE_DNORM (
REVENUEID CHAR(9) NOT NULL,
DRIVERID CHAR(9),
REVENUEDATE DATE,
GROSSTAKINGS NUMBER(8,2),
PAIDTOCOMPANY NUMBER(8,2),
NETEARNINGS NUMBER(8,2)
);

CREATE TABLE EMPLOYEE_DNORM (
NINUMBER CHAR(9) NOT NULL,
ADDRESS VARCHAR2(255),
POSTCODE VARCHAR(20),
CITY VARCHAR(50),
FIRSTNAME VARCHAR(255),
LASTNAME VARCHAR(255),
PPHONENO CHAR(11),
SPHONENO CHAR(11),
SEX CHAR(1),
DOB DATE,
EMAIL VARCHAR2(255),
HIREDATE DATE,
DRIVERLICENSEID CHAR(9),
```

```
PAYTYPE VARCHAR(9),
PAYVALUE NUMBER(9),
SALARY NUMBER(8,2),
TAXCODE VARCHAR(20)
);

CREATE TABLE REGULAR_BOOKING_DNORM (
REGBOOKID CHAR(9) NOT NULL,
CLIENTID CHAR(9),
BOOKINGFREQUENCY VARCHAR(7),
STARTDATE DATE,
DURATION VARCHAR(10),
NOOFPASSENGERS  NUMBER(2)
);

CREATE TABLE CLIENTS_DNORM (
CLIENTID CHAR(9) NOT NULL,
CLIENTTYPE VARCHAR(15),
CONTACTFNAME VARCHAR(255),
CONTACTLNAME VARCHAR(255),
PPHONENO CHAR(11),
SPHONENO CHAR(11),
CORPERATIONNAME VARCHAR(50),
EMAIL VARCHAR2(400)
);

CREATE TABLE EMPLOYEE_INCIDENTS_DNORM (
INCIDENTID CHAR(9) NOT NULL,
NINUMBER CHAR(9),
INCIDENTTYPE VARCHAR2(20),
INCIDENTNOTES VARCHAR2(4000),
INCIDENTDATE DATE
);

/* INSERTS */

/* INSERT INTO BOOKING_DNORM VALUES
('BK0000001','EP0000001','EP0000002', NULL,NULL,TO_DATE('08-DEC-
2019','DD-MON-YYYY'),TO_DATE('12-NOV-2020','DD-MON-YYYY'), 'PKUP 1
STREET EP CD1 LONDON','DRPOF 1 STREET EP CD1 LONDON',3,'07900000000',
'CASH', TO_DATE('11-NOV-2020','DD-MON-YYYY'), 'RAY', 'JAY', NULL, NULL,
NULL, 50.05); */
```

```
INSERT INTO BOOKING_DNORM VALUES ('BK0000002','EP0000003','EP0000004',
'RB0000001','CL123456A',TO_DATE('07-DEC-2019','DD-MON-
YYYY'),TO_DATE('11-NOV-2020','DD-MON-YYYY'), 'PKUP 2 STREET EP CD2
LONDON','DRPOF 2 STREET EP CD2 LONDON',5,'07900000001', 'CARD',
TO_DATE('09-NOV-2020','DD-MON-YYYY'), 'BEN', 'KEN', '4444555566667777',
TO_DATE('19-JAN-2025','DD-MON-YYYY'), '123', 40.66);
INSERT INTO BOOKING_DNORM VALUES ('BK0000003','EP0000005','EP0000006',
'RB0000003','CL123456B',TO_DATE('05-OCT-2018','DD-MON-
YYYY'),TO_DATE('11-MAR-2019','DD-MON-YYYY'), 'PKUP 3 STREET EP CD3
LONDON','DRPOF 3 STREET EP CD3 CARDIFF',5,'07900000002',
'CARD',TO_DATE('05-OCT-2018','DD-MON-YYYY'), 'DAN', 'TAN',
'1111555566667777', TO_DATE('19-JAN-2022','DD-MON-YYYY'), '130', 140);
INSERT INTO BOOKING_DNORM VALUES ('BK0000004','EP0000007','EP0000008',
'RB0000002','CL123456A',TO_DATE('08-AUG-2016','DD-MON-
YYYY'),TO_DATE('02-JAN-2020','DD-MON-YYYY'), 'PKUP 4 STREET EP CD4
LONDON','DRPOF 4 STREET EP CD4 LONDON',1,'07900000003', 'CASH',
TO_DATE('11-NOV-2020','DD-MON-YYYY'), 'AYA', 'YA', NULL, NULL, NULL,
6);

/* */

INSERT INTO REGULAR_BOOKING_DNORM VALUES
('RB000001','CL123456A','MONTHLY',TO_DATE('11-NOV-2020','DD-MON-
YYYY'),'30 DAYS','50');
INSERT INTO REGULAR_BOOKING_DNORM VALUES
('RB000003','CL123456B','WEEKLY',TO_DATE('11-JAN-2011','DD-MON-
YYYY'),'30 DAYS','2');
INSERT INTO REGULAR_BOOKING_DNORM VALUES
('RB000002','CL123456A','DAILY',TO_DATE('10-NOV-2020','DD-MON-
YYYY'),'14 DAYS','4');

/* */

INSERT INTO SHIFT_DNORM VALUES ('SH123456A', 'EP123456A','Dan',
'Fcface', 'OPERATOR',TO_DATE('02-NOV-2020', 'DD-MON-YYYY'), 'MORNING');
INSERT INTO SHIFT_DNORM VALUES ('SH123456B', 'EP123456B', 'Ray',
'Rampatino', 'OPERATOR',TO_DATE('04-NOV-2020', 'DD-MON-YYYY'),
'AFTERNOON');
INSERT INTO SHIFT_DNORM VALUES ('SH123456C', 'EP123456C','Digby',
'Dadarino', 'DRIVER',TO_DATE('08-NOV-2020', 'DD-MON-YYYY'), 'NIGHT');
INSERT INTO SHIFT_DNORM VALUES ('SH123456D', 'EP123456D','Dan2',
'Fcface2', 'OPERATOR',TO_DATE('02-NOV-2020', 'DD-MON-YYYY'),
'MORNING');
```

```
INSERT INTO SHIFT_DNORM VALUES ('SH123456E', 'EP123456E', 'Ray2',
'Rampatino2', 'OPERATOR',TO_DATE('04-NOV-2020', 'DD-MON-YYYY'),
'AFTERNOON');
INSERT INTO SHIFT_DNORM VALUES ('SH123456F', 'EP123456F','Digby2',
'Dadarino2', 'DRIVER',TO_DATE('08-NOV-2020', 'DD-MON-YYYY'),
'MORNING');

/* */

INSERT INTO CARS_DNORM
VALUES('CR1234A','DV123456A',2013,52041,TO_DATE('02-JAN-2020','DD-MON-
YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS_DNORM
VALUES('CR1234B','DV123456B',2019,43214,TO_DATE('08-JUL-2020','DD-MON-
YYYY'),7,'IN FOR SERVICE','DV123456B');
INSERT INTO CARS_DNORM
VALUES('CR1234C','DV123456C',2018,23791,TO_DATE('02-MAY-2020','DD-MON-
YYYY'),3,'ROADWORTHY','COMPANY*');

/* */

INSERT INTO DRIVER_REVENUE_DNORM VALUES
('RV123456A','DV123456A',TO_DATE('01-JUL-2020','DD-MON-
YYYY'),3000,476,2524);
INSERT INTO DRIVER_REVENUE_DNORM VALUES
('RV123456B','DV123456B',TO_DATE('22-AUG-2020','DD-MON-
YYYY'),2000,426,1574);
INSERT INTO DRIVER_REVENUE_DNORM VALUES
('RV123456C','DV123456C',TO_DATE('1-JUN-2020','DD-MON-
YYYY'),3000,393,2607);

/* */

INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456A','43 OCEANA CLOSE','N16
7GK','LONDON','FABIEN','GARNER','07000000028',NULL,'M',TO_DATE('05-AUG-
1974','DD-MON-YYYY'),'EMAIL1@EMAIL.COM',TO_DATE('08-JAN-2020','DD-MON-
YYYY'),'DV123456A','FIXEDFULL', 36000, NULL, NULL );
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456D','12 PARKLANE
AVENUE','N2F
0FL','LONDON','BRYAN','NAISMITH','07000000025',NULL,'M',TO_DATE('03-
MAY-1979','DD-MON-YYYY'),'EMAIL2@EMAIL.COM',TO_DATE('02-MAR-2020','DD-
MON-YYYY'),NULL,NULL,NULL, 40000, '1250L' );
```

```
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456B','1 BERKIN STREET','B1
1DA','BIRMINGHAM','AMY','HALLOW','07000000023',NULL,'F',TO_DATE('31-
MAY-1993','DD-MON-YYYY'),'EMAIL3@EMAIL.COM',TO_DATE('01-JUN-2020','DD-
MON-YYYY'),'DV123456B','FIXEDTEMP', 24000, NULL, NULL );
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456C','7 LADBROKE GROVE','W11
8FS','LONDON','GARY','JOHNSON','07000000022',NULL,'M',TO_DATE('25-JUL-
1992','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-
YYYY'),'DV123456C','FIXEDFULL', 36000, NULL, NULL );
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456D','7 LADBROKE GROVE','W11
8FS','LONDON','DARY',SON','07000000027',NULL,'M',TO_DATE('25-JUL-
1992','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-
YYYY'),'DV123456C','FIXEDFULL', 36000, NULL, NULL );
INSERT INTO EMPLOYEE_DNORM VALUES ('EP123456E','7 LADBROKE GROVE','W11
8FS','LONDON','RARY','ROHNSON','07000000026',NULL,'M',TO_DATE('25-JUL-
1992','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-NOV-2020','DD-MON-
YYYY'),'DV123456C','FIXEDFULL', 36000, NULL, NULL );

/* */

INSERT INTO EMPLOYEE_INCIDENTS_DNORM VALUES
('IN123456A','EP123456A','SICK',NULL,TO_DATE('02-NOV-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_DNORM VALUES
('IN123456B','EP123456B','DISCIPLINARY ACTION','BROUGHT ILLEGAL
SUBSTANCES INTO THE WORKPLACE',TO_DATE('06-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS_DNORM VALUES
('IN123456C','EP123456C','VACATION REQUEST','I HEAR SPAIN IS A GREAT
PLACE TO FIND CARS, MIND IF I TAKE A LOOK MYSELF?',TO_DATE('26-NOV-
2020','DD-MON-YYYY'));

/* */

INSERT INTO CLIENTS_DNORM VALUES
('CL123456A','PRIVATE','ALPHONSO','ROGERS','07500000013','07500000098',
NULL,'CLEMAIL1@EMAIL.COM');
INSERT INTO CLIENTS_DNORM VALUES
('CL123456B','CORPORATE','LIONEL','ALLAN','07500000012','07500000067','
PATEK INDUSTRIES','CLEMAIL2@EMAIL.COM');
INSERT INTO CLIENTS_DNORM VALUES
('CL123456C','PRIVATE','JAMIE','DAVIES','07500000011','07500000021',NUL
L,'CLEMAIL3@EMAIL.COM');

/* BIG TABLES */
```

```
DROP TABLE REGULAR_BOOKING_DNORM_BIG;
DROP TABLE CLIENTS_DNORM_BIG;
DROP TABLE CARS_DNORM_BIG;
DROP TABLE DRIVER_REVENUE_DNORM_BIG;
DROP TABLE BOOKING_DNORM_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_DNORM_BIG;
DROP TABLE SHIFT_DNORM_BIG;
DROP TABLE EMPLOYEE_DNORM_BIG;

CREATE TABLE REGULAR_BOOKING_DNORM_BIG AS SELECT * FROM
REGULAR_BOOKING_DNORM;
CREATE TABLE CLIENTS_DNORM_BIG AS SELECT * FROM CLIENTS_DNORM;
CREATE TABLE CARS_DNORM_BIG AS SELECT * FROM CARS_DNORM;
CREATE TABLE DRIVER_REVENUE_DNORM_BIG AS SELECT * FROM
DRIVER_REVENUE_DNORM;
CREATE TABLE BOOKING_DNORM_BIG AS SELECT * FROM BOOKING_DNORM;
CREATE TABLE EMPLOYEE_INCIDENTS_DNORM_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS_DNORM;
CREATE TABLE SHIFT_DNORM_BIG AS SELECT * FROM SHIFT_DNORM;
CREATE TABLE EMPLOYEE_DNORM_BIG AS SELECT * FROM EMPLOYEE_DNORM;

/* INCREASE EXPONENTIALLY */

SET SERVEROUTPUT ON;

DECLARE
cntr NUMBER := 0;
BEGIN
    FOR loop_counter IN 1..13 /*17*/
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_DNORM_BIG SELECT * FROM BOOKING_DNORM_BIG;
INSERT INTO REGULAR_BOOKING_DNORM_BIG SELECT * FROM
REGULAR_BOOKING_DNORM_BIG;
INSERT INTO SHIFT_DNORM_BIG SELECT * FROM SHIFT_DNORM_BIG;
INSERT INTO CARS_DNORM_BIG SELECT * FROM CARS_DNORM_BIG;
INSERT INTO DRIVER_REVENUE_DNORM_BIG SELECT * FROM
DRIVER_REVENUE_DNORM_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_DNORM_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_DNORM_BIG;
INSERT INTO CLIENTS_DNORM_BIG SELECT * FROM CLIENTS_DNORM_BIG;
INSERT INTO EMPLOYEE_DNORM_BIG SELECT * FROM EMPLOYEE_DNORM_BIG;
```

```
END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/

/* TABLE COUNTS */

/*
SELECT COUNT(*) FROM BOOKING_DNORM;
SELECT COUNT(*) FROM REGULAR_BOOKING_DNORM;
SELECT COUNT(*) FROM SHIFT_DNORM;
SELECT COUNT(*) FROM CARS_DNORM;
SELECT COUNT(*) FROM DRIVER_REVENUE_DNORM;
SELECT COUNT(*) FROM EMPLOYEE_INCIDENTS_DNORM;
SELECT COUNT(*) FROM CLIENTS_DNORM;

SELECT COUNT(*) FROM BOOKING_DNORM_BIG;
SELECT COUNT(*) FROM REGULAR_BOOKING_DNORM_BIG;
SELECT COUNT(*) FROM SHIFT_DNORM_BIG;
SELECT COUNT(*) FROM CARS_DNORM_BIG;
SELECT COUNT(*) FROM DRIVER_REVENUE_DNORM_BIG;
SELECT COUNT(*) FROM EMPLOYEE_INCIDENTS_DNORM_BIG;
SELECT COUNT(*) FROM CLIENTS_DNORM_BIG;
*/

/* DROP STATEMENTS */
DROP TABLE RECURRENT_BOOKING;
DROP TABLE PAYMENT;
DROP TABLE CORPORATE_CLIENT;
DROP TABLE REGULAR_BOOKING;
DROP TABLE CLIENTS;
DROP TABLE CARS;
DROP TABLE DRIVER_REVENUE;
DROP TABLE BOOKING;
DROP TABLE DRIVER;
DROP TABLE PAY_TYPE;
DROP TABLE OPERATOR;
DROP TABLE EMPLOYEE_INCIDENTS;
DROP TABLE SHIFT;
DROP TABLE EMPLOYEE;
DROP TABLE CITY_LOOKUP;

/* CITY_LOOKUP Table */
```

```
CREATE TABLE CITY_LOOKUP(
POSTCODE VARCHAR(8) PRIMARY KEY,
CITY VARCHAR(50)
);

/* CLIENTS Table */
CREATE TABLE CLIENTS (
CLIENTID CHAR(9) PRIMARY KEY,
CLIENTTYPE VARCHAR(15),
CONTACTFNAME VARCHAR(255),
CONTACTLNAME VARCHAR(255),
PPHONENO CHAR(11),
SPHONENO CHAR(11),
EMAIL VARCHAR2(400)
);

/* PAY_TYPE Table */
CREATE TABLE PAY_TYPE (
PAYTYPE VARCHAR(9) PRIMARY KEY,
PAYVALUE NUMBER(9)
);

/* EMPLOYEE Table */
CREATE TABLE EMPLOYEE(
NINUMBER CHAR(9) PRIMARY KEY,
POSTCODE VARCHAR(20),
ADDRESS VARCHAR2(255),
FIRSTNAME VARCHAR(255),
LASTNAME VARCHAR(255),
PPHONENO CHAR(11),
SPHONENO CHAR(11),
SEX CHAR(1),
DOB DATE,
EMAIL VARCHAR2(255),
HIREDATE DATE
);

/* OPERATOR Table */
CREATE TABLE OPERATOR (
OPERATORID CHAR(9),
EMPLOYEEID CHAR(9),
SALARY NUMBER(8,2),
TAXCODE VARCHAR(20)
```

```
);

/* DRIVER Table */
CREATE TABLE DRIVER (
DRIVERLICENSEID CHAR(9),
EMPLOYEEID CHAR(9),
PAYTYPE VARCHAR (9)
);

/* BOOKING Table */
CREATE TABLE BOOKING (
BOOKINGID CHAR(9) PRIMARY KEY,
OPERATORID CHAR(9),
DRIVERID CHAR(9),
DATEBOOKED DATE,
PICKUPDATE DATE,
PICKUPADDRESS VARCHAR(255),
DROPOFFADDRESS VARCHAR2(255),
NOOFPASSENGERS NUMBER(2),
CUSTOMERPHONENUMBER CHAR(11)
);

/* REGULAR_BOOKING Table */
CREATE TABLE REGULAR_BOOKING(
REGBOOKID CHAR(9) PRIMARY KEY,
CLIENTID CHAR(9),
BOOKINGFREQUENCY VARCHAR(50),
STARTDATE DATE,
NOOFTRANSPORTEES NUMBER(2),
DURATION VARCHAR(10)
);

/* DRIVER_REVENUE Table */
CREATE TABLE DRIVER_REVENUE(
REVENUEID CHAR(9) PRIMARY KEY,
DRIVERID CHAR(9),
REVENUEDATE DATE,
GROSSTAKINGS NUMBER(8,2),
PAIDTOCOMPANY NUMBER(8,2),
NETEARNINGS NUMBER(8,2)
);

/* CORPERATE_CLIENT Table */
```

```
CREATE TABLE CORPORATE_CLIENT (
CLIENTID CHAR(9) PRIMARY KEY,
CORPERATIONNAME VARCHAR(50)
);

/* EMPLOYEE_INCIDENTS Table */
CREATE TABLE EMPLOYEE_INCIDENTS (
INCIDENTID CHAR(9) PRIMARY KEY,
NINUMBER CHAR(9),
INCIDENTTYPE VARCHAR2(20),
INCIDENTNOTES VARCHAR2(4000),
INCIDENTDATE DATE
);

/* CARS Table */
CREATE TABLE CARS (
REGISTRATIONNO CHAR(7) PRIMARY KEY,
DRIVERID CHAR(9),
REGISTRATIONYEAR INTEGER,
MILEAGE VARCHAR(10),
LASTMOTDATE DATE,
NOOFSEATS VARCHAR(10),
CARSTATUS VARCHAR(30),
OWNERID VARCHAR(9)
);

/* SHIFT Table */
CREATE TABLE SHIFT (
SHIFTID CHAR(9) PRIMARY KEY,
EMPLOYEEID CHAR(9),
SHIFTDATE DATE,
SHIFTTYPE VARCHAR(20)
);

/* PAYMENT Table */
CREATE TABLE PAYMENT (
PAYMENTID CHAR(9) PRIMARY KEY,
BOOKINGID CHAR(9),
PAYMENTMETHOD VARCHAR(10),
PAYMENTDATE DATE,
PAYEESFIRSTNAME VARCHAR(255),
PAYEESLASTNAME VARCHAR(255),
CARDNUMBER VARCHAR(16),
```

```
EXPIRYDATE DATE,
CVV CHAR(3),
AMOUNTPAID NUMBER(8,2)
);

/* RECURRENT_BOOKING Table */
CREATE TABLE RECURRENT_BOOKING (
BOOKINGID CHAR(9),
REGBOOKID CHAR(9)
);
/* CITY_LOOKUP Inserts */
INSERT INTO CITY_LOOKUP VALUES ('N14 4AU', 'NORTH LONDON');
INSERT INTO CITY_LOOKUP VALUES ('NW8 9ZL', 'NORTH WEST LONDON');
INSERT INTO CITY_LOOKUP VALUES ('SW17 6AF', 'SOUTH WEST LONDON');

/* EMPLOYEE Inserts */
INSERT INTO EMPLOYEE VALUES ('EP123456A','B1 1DA','5 ALLINGTON
ROAD','Dan', 'Fcface','07000000000',NULL,'M',TO_DATE('23-JAN-2002','DD-
MON-YYYY'),'EMAIL1@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456B','B1 1DA','44 ALLINGTON ROAD',
'Ray', 'Rampatino','07000000001','07000000002','M',TO_DATE('12-JAN-
1998','DD-MON-YYYY'),'EMAIL2@EMAIL.COM',TO_DATE('18-MAY-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456C','B1 1DA','39 ALLINGTON
ROAD','Digby','Dadarino','07000000003',NULL,'M',TO_DATE('23-JAN-
2002','DD-MON-YYYY'),'EMAIL3@EMAIL.COM',TO_DATE('06-NOV-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456D','B1 1DA','60 ALLINGTON
ROAD','Dan2','Fcface2','07000000004',NULL,'M',TO_DATE('12-FEB-
2001','DD-MON-YYYY'),'EMAIL4@EMAIL.COM',TO_DATE('05-FEB-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456E','B1 1DA','90 ALLINGTON
ROAD','Ray2', 'Rampatino2','07000000005',NULL,'M',TO_DATE('12-JAN-
1971','DD-MON-YYYY'),'EMAIL5@EMAIL.COM',TO_DATE('01-JUN-2020','DD-MON-
YYYY'));
INSERT INTO EMPLOYEE VALUES ('EP123456F','N14 4AU','63 KEYSE
ROAD','Digby2',
'Dadarino2','07000000006','07000000007','M',TO_DATE('16-OCT-1978','DD-
MON-YYYY'),'EMAIL6@EMAIL.COM',TO_DATE('05-MAR-2020','DD-MON-YYYY'));

/* CLIENTS Inserts */
```

```
INSERT INTO CLIENTS VALUES
('CL123456A','PRIVATE','EDWARD','ELRIC','07500000000',NULL,'CLEMAIL1@EM
AIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456B','PRIVATE','HASHIRAMA','SENJU','07500000001',NULL,'CLEMAIL2
@EMAIL.COM');
INSERT INTO CLIENTS VALUES
('CL123456C','PRIVATE','RINTAROU','OKABE','07500000002','07500000022','
CLEMAIL3@EMAIL.COM');

/* PAY_TYPE Inserts */
INSERT INTO PAY_TYPE VALUES ('FIXEDTEMP', 24000);
INSERT INTO PAY_TYPE VALUES ('FIXEDFULL', 36000);
INSERT INTO PAY_TYPE VALUES ('COMM75', 75);

/* OPERATOR Inserts */
INSERT INTO OPERATOR VALUES('OP123456A','EP123456A',40000,'1250L');
INSERT INTO OPERATOR VALUES('OP123456B','EP123456B',42500,'1250L');
INSERT INTO OPERATOR VALUES('OP123456C','EP123456C',39000,'1250L');

/* DRIVER Inserts */
INSERT INTO DRIVER VALUES ('DV123456A','EP123456D','FIXEDFULL');
INSERT INTO DRIVER VALUES ('DV123456B','EP123456E','COMM75');
INSERT INTO DRIVER VALUES ('DV123456C','EP123456F','FIXEDFULL');

/* BOOKING Inserts */
INSERT INTO BOOKING VALUES
('BK0000002','OP123456A','DV123456A',TO_DATE('08-NOV-2020','DD-MON-
YYYY'),TO_DATE('12-NOV-2020','DD-MON-YYYY'),'21 BALFE STREET','79
TEMPLE WAY',3,'07900000000');
INSERT INTO BOOKING VALUES
('BK0000003','OP123456B','DV123456B',TO_DATE('07-MAY-2020','DD-MON-
YYYY'),TO_DATE('08-MAY-2020','DD-MON-YYYY'),'78 TRAILLE STREET','76
BOAR LANE',5,'07900000001');
INSERT INTO BOOKING VALUES
('BK0000004','OP123456C','DV123456C',TO_DATE('15-APR-2020','DD-MON-
YYYY'),TO_DATE('15-APR-2020','DD-MON-YYYY'),'72 QUAY STREET','63 PRINCE
CONSORT ROAD',5,'07900000002');

/* REGULAR_BOOKING Inserts */
INSERT INTO REGULAR_BOOKING VALUES
('RB0000001','CL123456A','WEEKLY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),3,'7 WEEKS');
```

```
INSERT INTO REGULAR_BOOKING VALUES
('RB0000002','CL123456B','DAILY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),5,'4 DAYS');
INSERT INTO REGULAR_BOOKING VALUES
('RB0000003','CL123456A','WEEKLY',TO_DATE('02-NOV-2020','DD-MON-
YYYY'),5,'4 WEEKS');

/* DRIVER_REVENUE Inserts */
INSERT INTO DRIVER_REVENUE VALUES ('RV123456A','DV123456A',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,476,2524);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456B','DV123456B',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),2932,833,2099);
INSERT INTO DRIVER_REVENUE VALUES ('RV123456C','DV123456C',TO_DATE('01-
NOV-2020','DD-MON-YYYY'),3000,497,2503);

/* CORPERATE_CLIENT Inserts */
INSERT INTO CORPORATE_CLIENT VALUES('CL123456A','UMBRELLA CORP');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456B','WONKA INDUSTRIES');
INSERT INTO CORPORATE_CLIENT VALUES('CL123456C','STARK INDUSTRIES');

/* EMPLOYEE_INCIDENTS Inserts */
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456A','EP123457A','SICK','GOT MY FINGERS STUCK IN A BOWLING
BALL',TO_DATE('02-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456B','EP123456B','VACATION REQUEST','I WANT TO GO ON MY
HONEYMOON',TO_DATE('10-NOV-2020','DD-MON-YYYY'));
INSERT INTO EMPLOYEE_INCIDENTS VALUES
('IN123456C','EP123456C','DISCIPLINARY ACTION','CAME TO WORK DRUNK.
LICENSE SUSPENDED UNTIL FURTHER NOTICE',TO_DATE('02-OCT-2020','DD-MON-
YYYY'));
INSERT INTO CARS VALUES('CR1234A','DV123456A',2018,52041,TO_DATE('06-
AUG-2020','DD-MON-YYYY'),4,'ROADWORTHY','COMPANY*');
INSERT INTO CARS VALUES('CR1234B','DV123456B',2008,112218,TO_DATE('28-
AUG-2020','DD-MON-YYYY'),4,'AWAITING REPAIR','COMPANY*');
INSERT INTO CARS VALUES('CR1234C','DV123456C',2012,178627,TO_DATE('28-
APR-2020','DD-MON-YYYY'),5,'ROADWORTHY','DV123456C');

/* SHIFT Inserts */
INSERT INTO SHIFT VALUES ('SH123456A','EP123456A',TO_DATE('05-NOV-
2020','DD-MON-YYYY'),'MORNING');
INSERT INTO SHIFT VALUES ('SH123456B','EP123456B',TO_DATE('25-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
```

```
INSERT INTO SHIFT VALUES ('SH123456C','EP123456C',TO_DATE('29-NOV-
2020','DD-MON-YYYY'),'NIGHT');
INSERT INTO SHIFT VALUES ('SH123456D','EP123456D',TO_DATE('05-NOV-
2020','DD-MON-YYYY'),'MORNING');
INSERT INTO SHIFT VALUES ('SH123456E','EP123456E',TO_DATE('13-NOV-
2020','DD-MON-YYYY'),'AFTERNOON');
INSERT INTO SHIFT VALUES ('SH123456F','EP123456F',TO_DATE('08-NOV-
2020','DD-MON-YYYY'),'MORNING');

/* PAYMENT Inserts */
INSERT INTO PAYMENT VALUES('PY123456A','BK0000002','CARD',TO_DATE('08-
NOV-2020','DD-MON-
YYYY'),'KAZUTO','KIRIGAYA','4432491216876558',TO_DATE('AUG-2023','MON-
YYYY'),'296',40.66);
INSERT INTO PAYMENT VALUES('PY123456B','BK0000003','CARD',TO_DATE('07-
MAY-2020','DD-MON-
YYYY'),'KURISU','MAKISE','5127071833675349',TO_DATE('AUG-2023','MON-
YYYY'),'314',140);
INSERT INTO PAYMENT VALUES('PY123456C','BK0000004','CARD',TO_DATE('15-
APR-2020','DD-MON-
YYYY'),'HACHIMAN','HIKIGAYA','52132116754210 49',TO_DATE('AUG-
2023','MON-YYYY'),'329',6);

/* RECURRENT_BOOKING Inserts */
INSERT INTO RECURRENT_BOOKING VALUES('BK0000002','RB0000001');
INSERT INTO RECURRENT_BOOKING VALUES('BK0000003','RB0000002');
INSERT INTO RECURRENT_BOOKING VALUES('BK0000004','RB0000003');



/* BIG TABLES (Normalized) */
DROP TABLE REGULAR_BOOKING_BIG;
DROP TABLE CLIENTS_BIG;
DROP TABLE CARS_BIG;
DROP TABLE DRIVER_REVENUE_BIG;
DROP TABLE BOOKING_BIG;
DROP TABLE EMPLOYEE_INCIDENTS_BIG;
DROP TABLE SHIFT_BIG;
DROP TABLE EMPLOYEE_BIG;

DROP TABLE RECURRENT_BOOKING_BIG;
DROP TABLE PAY_TYPE_BIG;
DROP TABLE DRIVER_BIG;
```

```
DROP TABLE OPERATOR_BIG;
DROP TABLE  CITY_LOOKUP_BIG;
DROP TABLE CORPORATE_CLIENT_BIG;
DROP TABLE PAYMENT_BIG;

CREATE TABLE REGULAR_BOOKING_BIG AS SELECT * FROM REGULAR_BOOKING;
CREATE TABLE CLIENTS_BIG AS SELECT * FROM CLIENTS;
CREATE TABLE CARS_BIG AS SELECT * FROM CARS;
CREATE TABLE DRIVER_REVENUE_BIG AS SELECT * FROM DRIVER_REVENUE;
CREATE TABLE BOOKING_BIG AS SELECT * FROM BOOKING;
CREATE TABLE EMPLOYEE_INCIDENTS_BIG AS SELECT * FROM
EMPLOYEE_INCIDENTS;
CREATE TABLE SHIFT_BIG AS SELECT * FROM SHIFT;
CREATE TABLE EMPLOYEE_BIG AS SELECT * FROM EMPLOYEE;

CREATE TABLE RECURRENT_BOOKING_BIG AS SELECT * FROM RECURRENT_BOOKING;
CREATE TABLE PAY_TYPE_BIG AS SELECT * FROM PAY_TYPE;
CREATE TABLE DRIVER_BIG AS SELECT * FROM DRIVER;
CREATE TABLE OPERATOR_BIG AS SELECT * FROM OPERATOR;
CREATE TABLE CITY_LOOKUP_BIG AS SELECT * FROM CITY_LOOKUP;
CREATE TABLE CORPORATE_CLIENT_BIG AS SELECT * FROM CORPORATE_CLIENT;
CREATE TABLE PAYMENT_BIG AS SELECT * FROM PAYMENT;

/* INCREASE EXPONENTIALLY (Normalized) */

SET SERVEROUTPUT ON;

DECLARE
cntr NUMBER := 0;
BEGIN
    FOR loop_counter IN 1..13 /*17*/
LOOP
cntr := cntr  + 1;
INSERT INTO BOOKING_BIG SELECT * FROM BOOKING_BIG;
INSERT INTO REGULAR_BOOKING_BIG SELECT * FROM REGULAR_BOOKING_BIG;
INSERT INTO SHIFT_BIG SELECT * FROM SHIFT_BIG;
INSERT INTO CARS_BIG SELECT * FROM CARS_BIG;
INSERT INTO DRIVER_REVENUE_BIG SELECT * FROM DRIVER_REVENUE_BIG;
INSERT INTO EMPLOYEE_INCIDENTS_BIG SELECT * FROM
EMPLOYEE_INCIDENTS_BIG;
INSERT INTO CLIENTS_BIG SELECT * FROM CLIENTS_BIG;

INSERT INTO RECURRENT_BOOKING_BIG SELECT * FROM RECURRENT_BOOKING_BIG;
```

```
INSERT INTO PAY_TYPE_BIG SELECT * FROM PAY_TYPE_BIG ;
INSERT INTO DRIVER_BIG SELECT * FROM DRIVER_BIG;
INSERT INTO CITY_LOOKUP_BIG SELECT * FROM CITY_LOOKUP_BIG;
INSERT INTO CORPORATE_CLIENT_BIG SELECT * FROM CORPORATE_CLIENT_BIG;
INSERT INTO OPERATOR_BIG  SELECT * FROM OPERATOR_BIG;
INSERT INTO PAYMENT_BIG SELECT * FROM PAYMENT_BIG;
INSERT INTO EMPLOYEE_BIG SELECT * FROM EMPLOYEE_BIG;

END LOOP;
dbms_output.put_line('Executed: ' || cntr) ;
END;
/

SET TIMING ON
```