AutoZone
(A National retail and service store)

Name: Naga Venkata Kanakalakshmi
UNT ID: 11725119
Project – Part 5: Normalization

Group Number: 8
Group Members:

Name	UNT ID	Mail ID
Naga Venkata kanakalakshmi	11725119	nagavenkatakanakalmurikipudi@my.unt.edu
Vishnu Vardhan Reddy Sudireddy	11642773	vishnusudireddy@my.unt.edu
Srinivas Sankula	11667743	srinivassankula@my.unt.edu
Sai Sindhu Rudraraju	11644475	saisindhurudraraju@my.unt.edu

Introduction:
AutoZone, a national retailer of automotive parts and accessories Company has requirement to store the information of the inventory, employees, insurance, servicing, payment plans, supplier details and customer details. AutoZone also want to store the information about locations, jobs done by the employees and automotive retailer branch details.

Updated Description: (Highlighted with red color is existing and blue color is new table after decomposition)

- Each retailer in the AutoZone chain needs to be identified based on the retailer id. It can have the basic information like contact, business hour and each AutoZone retailer can have a manager and the website details related to the that location. Multiple employees can work in the one automotive retailer. Each retailer can have in house inventory and external inventory. Automotive retailer address/location needs to be stored. Each automotive retailer can afford multiple jobs or services.
- The "automotive_retailer" table serves as a central database entity for vital retailer information. It includes attributes such as a unique manager identifier ("manager_id"), a primary key ("id"), a foreign key linking to "automotive_retailer_contact" ("phone"), "business_hours," and another foreign key referencing an address table ("address_id"). All attributes enforce a not-null constraint. The table establishes connections, including multiple associations with "employee," "inventory_product," and "job," as well as a singular connection with "address" and "automotive_retailer_contact." These linkages facilitate a dynamic relationship between the automotive retailer and associated entities within the database.

- Inventory product entity serves as a major component and uniquely identified in this system. It manages the various automotive products. It includes the attributes like name, quantity, price, automotive_retailer_id, automobile_id, address_id. It holds the details of the automotive retailer, automobile data. It establishes the relation with suppliers, automotive retailers, part service, automobile, address and bill.It serves as a central component for efficient inventory management, procurement, and tracking within the system.
- Employee entity holds the data of the employees working in AutoZone. Each employee is uniquely identified with their ID. Along with the ID this entity will also have the attributes first name, last name, date of birth, phone number, email address, annual salary, ssn, address, hire date and the automotive retailer ID. An employee can create many bills for the customers and so the employee entity will form a one-to-many relation with the bill entity. More than one employee can have the same address as there is a chance of 2 employees living in the same location. So, employee will form a many to one relationship with the address entity. Many employees can be part of a job and one employee can be part of many jobs. So, employee entity will form many to many relationships with the job entity. As Many employees work in one location, employee entity will form a many to one relationship with the automotive retailer entity. Employees may receive multiple paychecks over a period. So, Employee entity will form a one-to-many relationship with employee payroll entity.
- Each employee is uniquely identified by their "Id" and has essential info like their "Email," "Annual Salary," and "Hire Date." There's a personal touch with the "SSN" attribute, ensuring a one-to-one relationship with social security numbers. When it comes to where they work, the table reflects that each employee is connected to one automotive retailer, forming a kind of work family, and it goes both ways – an automotive retailer can have several employees. Additionally, the table shows that each employee has a link to an address, recognizing that sometimes colleagues might share the same location. These changes aim to keep things clear and straightforward.
- Employee payroll entity has the record of paychecks given to employees. Each pay given to an employee is identified by a unique id. This entity has the attributes hours worked (number of hours worked by employee during the pay cycle), start date (start date of the pay cycle), end date (end date of the pay cycle), pay (amount paid to the employee) and employee id (id of the employee that is used to uniquely identify an employee). The employee payroll entity will form a many to one relationship with the employee as one employee may receive many pays over a period.
- AutoZone can have suppliers to inventory who supply the products that stored or used during the services. An inventory can have multiple suppliers and vice versa. Suppliers' information like contact and address needs to be stored. Suppliers can uniquely be identified by their id.
- Address entity serves as a global component within the system, defines location information. It is uniquely identified and includes the attributes like apartment number, street, city, state, country, and ZIP code, all of them are mandatory. This entity forms

relationships with other entities in various ways: many employees may have a single address, inventory products are associated with specific addresses, automotive retailers and suppliers have distinct addresses, and multiple customers can be linked to one address.

- Bill entity is crucial for recording financial transactions. It is uniquely defined and includes attributes like date, payment mode, insurance, customer, job, employee, sale type (can be an online or offline), and payment plan. It forms relationships with various entities: multiple bills can be linked to one employee, have many-to-many connections with inventory products, and maintain multiple bills of each association with insurance, customers, and payment plans. Additionally, every job has a specific bill.
- Job entity is essential for managing automotive service tasks. It is uniquely defining and includes attributes like date, description, customers, automotive retailers, VIN numbers, and automobiles. All of which are mandatory. "Job" forms key relationships: It establishes a direct link with billing records, multiple jobs are associated with automobiles and automotive retailers, Customers. Various jobs can be done by different employees. Whereas Single job can be performed using multiple part services. This entity serves as a central hub for tracking and managing automotive service jobs. Job table also maintains job_status which describes about the status of the job such as ‘completed’, ‘inprogress’, ‘unassigned’.
- The "Job" entity, crucial for managing automotive service tasks, is uniquely defined and comprises mandatory attributes such as date, description, VIN numbers, and automotive retailer details. It forms key relationships with billing records, multiple jobs linked to automobiles and automotive retailers, and various employees performing different jobs. Additionally, the table tracks job status, indicating whether a job is 'completed,' 'in progress,' or 'unassigned.'
- Part service like during the service which are automotive parts we are using to get the job done for a vehicle. It may have the information like job details, name/description of the service and it can have quantity of the parts which are using for that service. Part service can be identified by their id. Part service can have type like is it only service or any parts used to get that service done. In this many parts service can be related to one job and it have information related to the inventory products to know the parts related to the which inventory.
- Customer entity represents individuals in the system and is identified uniquely. It includes essential attributes like first name, last name, birthdate, driver's license, phone, and address. Multiple customers may opt for multiple insurance policies. A Customer can be associated with multiple jobs which has multiple bills. Many customers may have the single address. This entity enables efficient management of customer data, service history, and financial transactions.
- The "Customer" entity, representing individuals uniquely in the system, includes essential attributes like driver's license, phone, and address. Multiple customers can opt for multiple insurance policies, and a customer can be associated with multiple jobs, each having multiple bills. Many customers may share a single address. This streamlined

entity ensures efficient management of customer data, service history, and financial transactions, with a focus on key attributes like driver's license, phone, and address.

- Insurance is like if customer wants to utilize his insurance plan for the payment, we need to store the information of that insurance. It can have policy type, provider and claim percentage. The particular insurance plan can be identified by plan id. A customer can have multiple insurance and vice versa. we would also like to include insurance id in the bill, it will be like one insurance plan can be included in many bills.
- Payment plan like if customer interested in the paying in instalments, he can use this option. This payment plan will have the information like plan name, number of instalments and the interest rate related to the payment plan. Each payment can be identified by plan id and multiple bills can have single payment plan.
- Would like to store the information about automobile. So that when the job is performed, we can use the parts related to the specific automobile model. It can have the information like manufacture, name, variant, year of the build and color of the vehicle. Each automobile model can be identified by automobile id. An automobile can have multiple products in the inventory and would like to keep the automobile information in the jobs performed. It will be like multiple jobs can be performed to an automobile.
- Inventory-product-supplier is a relation table. Which has a primary and foreign key as inventory_product_id, supplier_id. It has a single attribute quantity. Multiple suppliers will provide multiple products. In this we maintain the product_id, supplier_id and the product quantity provided by the supplier.
- Bill_inventory_product is a relation table. Which has a primary and foreign key as bill_id, inventory_product_id. It has a single attribute quantity. Multiple products will have the multiple bills. In this we maintain the bill_id, product_id and the product quantity in each generated bill.
- Job_employee is a relation table. Which has the primary and foreign key as job_id and employee_id. There are multiple employees, working for the on the different jobs to maintains the those details this table helps to track the respective details. It’s a join of employee and job and the job table.
- Customer_insurance is a relation table. Which has the primary and foreign key as cutomer_id and insurance_id. It has a single attribute status. Many customers can opt for multiple insurances. This table helps to track the insurance opted by the customers and their status.
- In the automotive_retailer a name attribute is newly added. The name identifier helps to track the name of specific automotive_retailer.

Below is the description of new tables which are formed as part of Normalization:

- The newly added "SSN" table introduces a set of personal details crucial for employee identification and recognition. This table includes attributes such as "ssn" (Social Security Number), "first_name," "last_name," "dob" (date of birth), "gender," and "phone." The purpose of this table is to store individualized information, creating a comprehensive

profile for each employee. The "SSN" attribute establishes a one-to-one relationship with the "Employee" table, emphasizing the unique link between an employee and their Social Security Number. This design ensures that each employee's personal information is securely and uniquely tied to their employment records, enhancing data organization and privacy. In essence, the "SSN" table enriches the employee database by incorporating detailed personal attributes while maintaining a seamless one-to-one relationship with the core employee information.

- The "automotive_retailer_contact" table captures contact details for automotive retailers. It includes attributes such as "PHONE," "NAME," "EMAIL," and "WEBSITE." The "PHONE" attribute serves as the primary key for unique identification. This table is designed to efficiently manage contact information, facilitating communication and information retrieval for automotive retailers in the database.
- The "customer_driverlicense" table is designed to store information related to customers and their driver's licenses. It includes attributes such as "first_name," "last_name," "dob" and "DRIVERLICENSE," which serves as the primary key for unique identification. This table facilitates the efficient management of customer details, including names, date of birth, and driver's license information, in the database.
- The "Job_Customer" table serves to establish connections between automotive service jobs, customers, and automobiles. It comprises attributes such as "vin_number" (primary key), "customer_id," and "automobile_id." The "vin_number" uniquely identifies each record, while "customer_id" and "automobile_id" are foreign keys linking to the "customer" and "automobile" tables, respectively. This table facilitates the systematic tracking and management of automotive service jobs in relation to specific customers and their automobiles in the database.

Assumptions:

- A product can be available both in store and in warehouse. If the product has automative_retailer_id same as address id, that means the product is in store. And if automative_retailer_id is null, then the inventory is not in store.
- Automobile entity will have the generic information about the model, manufacturer and make of the automobile available in the market.
- A job will be created if a customer wants to get his automobile serviced in the store and the bill will be generated for the job.
- A bill can be generated without a job when a customer purchases products from the store without getting the service done in store where this data is stored in bill_inventory table.
- Employee can receive multiple payments over a period. This is maintained in employee_payroll table.
- Employee can be part of the job and he can generate bill for the job. And employee can sell the products to the customer who is not seeking service (not part of the job) from the retailer.

- Address is global table where the addresses of suppliers, customers, employees, inventory and retailer are stored.
- customer can pay the bill directly or he can opt for payment plan. Customer can select the payment plan form the payment_plan table.
- Job will be created if a customer opts services from the retailer. services are work done by the employees on the automobile to fix issues or install accessories.
- Part_services have the information about the products used in the job. One Job can have multiple products.
- Insurance table has different insurance policies that are available in the market.
- more than one person (customer or employee) can have same address. but no 2 suppliers and retailers can have the same address.
- product id in part service can be null as labor chargers of a job can also be clocked in part service where no inventory will be tagged to that service.
- The total amount of the bill of a customer who opted for a payment plan will be recorded as a transaction in bill table. And the installments are also saved as transactions in the same table.
- A phone number must consist of 10 digits and should only contain numerical characters, with no special symbols or spaces.
- Each email address must be unique, meaning no two users can share the same email ID.
- In the bill table, the sale type can be categorized as either online or offline.
- The vin_number in the job table must have a fixed length of 17 characters.
- The quantity of one item in a bill cannot exceed 10, as customers are limited to purchasing a maximum of 10 identical items.
- The status of insurance in the customer_insurance table can be either active or inactive.
- Zip codes are required to have a fixed length of 5 characters.
- Employees cannot work for more than 100 hours within a two-week period.
- The mode of payment in the bill table can be either cash or card.
- Sales cannot occur across different AutoZone locations. It is limited to few locations.
- When an automotive retailer is deleted from a location, the suppliers associated with that location are also removed.
- In our autozone, in-progress jobs play the vital role until the job_satus changed to completed. so, any changes on the customer table are closely monitored.

Normalization:

- *Normalization is a process of organizing data in a database to reduce redundancy and dependency. It involves breaking down a database into multiple related tables and establishing relationships between them. This helps to eliminate data redundancy and ensures that the data is stored in a structured and efficient manner.*
- *Initially, we have 17 tables to represent "AutoZone" (A National retail and service store), after keen evaluation, we have found that 4 tables is in second Normal Form and rest are in BCNF. So, we have applied the LLJD-BCNF Algorithm to decompose the respective tables and make the decomposed tables loss less, dependency preserving, and in BCNF.*

- *Now, we will discuss how we have found the tables is in 2NF and later, how we have decomposed it to BCNF. Please find the tables below, which has the information of how we have found the tables is in 2NF and the tables has the latest tables after normalization to BCNF too.*

Auto Zone Tables Normalization

Tables	Attributes=short Forms	FD's	Closure	Candidate Key	Prime	non-prime	super keys	Highest Normal Form	Justification
automotive_retailer	id = A (primary Key) phone = B email = C website = D business_hours = E manager_id = F address_id = G name = H	(A -> B, A -> E, A -> F, A -> G, B -> H, B -> C, B -> D)	A+ = ABCDEFGH B+ = BCDH C+ = C D+ = D E+ = E F+ = F G+ = G H+ = H	A	A	BCDEFGH	A,AB,AC,AD,AE,AF,AG,AH,ABC,ABD,ABE,ABF,ABG,ABH,ACDE,ACDG,ADEFG,ABCDEFG, ABCDEFGH, so on..	The Table is is 2NF	*Not BCNF - The left-hand side of each FD in FD set is not a super key * Not 3NF - The right-hand of each FD in FD set is not a prime attribute. * It is 2NF - Every non-prime attribute is fully dependent on candidate key
inventory_product	id = A name = B quantity = C price = D automotive_retailer_id = E automobile_id = F address_id = G	A ->B, A -> C, A -> D, A -> E, A -> F, A -> G	A+ = ABCDEFG B+ = B C+ = C D+ = D E+ = E F+ = F G+ = G	A	A	B, C, D, E, F, G,	A, AB, AC, AD, AE, AF, AG, ABC, ABD, ABE, ABF, ABG, ACD, ACE, ACF, ACG, ADE, ADF, ADG, AEF, AEG, AFG, ABCD, ABCE, ABCF, ABCG, ABDE, ABDF, ABDG, ABEF, ABEG, ABFG, ACDE, ACDF, ACDG, ACEF, ACEG, ACFG, ADEF, ADEG, ADFG, AEFG, ABCDE, ABCDF, ABCDG, ABCEF, ABCEG, ABCFG, ABDEF, ABDEG, ABDFG, ABEFG, ACDEF, ACDEG, ACDFG, ACEFG, ADEFG, ABCDEF, ABCDEG, ABCDFG, ABCEFG, ABDEFG, ACDEFG, ABCDEFG	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
automobile	id = A (Primary Key) manufacturer = B name = C variant = D year = E color = F	(A -> B, A -> C, A -> D, A ->E, A -> F)	A+ = ABCDEF B+ = B C+ = C D+ = D E+ = E F+ = F	A	A	B, C, D, E, F	A, AB, AC, AD, AE, AF, ABC, ABD, ABE, ABF, ACD, ACE, ACF, ADE, ADF, AEF, ABCD, ABCE, ABCF, ABDE, ABDF, ABEF, ACDE, ACDF, ACEF, ADEF, ABCDE, ABCDF, ABCEF, ABDEF, ACDEF, ABCDEF	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
supplier	id = A (Primary Key) name = B phone = C address_id = D	(A -> B, A -> C, A -> D)	A+ = ABCD B+ = B C+ = C D+ =D	A	A	B, C, D	A, AB, AC, AD, ABC, ABD, ACD, ABCD	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
Employee	id = A (primary Key) first_name = B last_name = C dob = D phone = E email = F annual_salary = G ssn = H automotive_retailer_id = I address_id = J hire_date = K	(A -> F, A -> G, A -> H, A -> I, A -> J, A -> K H -> E, H -> D, H -> B, H -> C)	A+ = AFGHIJK B+ = B C+ = C D+ = D E+ = E F+ = F G+ = G H+ =HEDBC I+ = I J+ = J K+ = K	A	A	B, C, D, E, F, G, H, I, J, K	A,AB,AC,AD,AE,AF,AG,AH,AI,AJ,AK,ABC,ABD,ABE,ABF,ABG,ABH,ABI,ABJ,ABK so on..	The Table is is 2NF	*Not BCNF - The left-hand side of each FD in FD set is not a super key * Not 3NF - The right-hand of each FD in FD set is not a prime attribute. * It is 2NF - Every non-prime attribute is fully dependent on candidate key
employee_payroll	id = A (Primary Key) hours_worked = B start_date = C end_date = D pay = E employee_id = F	(A -> B, A -> C, A -> D, A -> E, A -> F)	A+ = ABCDEF B+ = B C+ = C D+ = D E+ = E F+ = F	A	A	B, C, D, E, F	A, AB, AC, AD, AE, AF, ABC, ABD, ABE, ABF, ACD, ACE, ACF, ADE, ADF, AEF, ABCD, ABCE, ABCF, ABDE, ABDF, ABEF, ACDE, ACDF, ACEF, ADEF, ABCDE, ABCDF, ABCEF, ABDEF, ACDEF, ABCDEF	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
address	id = A (primary Key) apartment_no = B street = C city = D state = E country = F zip = G	(A -> B, A -> C, A -> D, A-> E, A-> F, A -> G)	A+ = ABCDEFG B+ = B C+ = C D+ = D E+ = E F+ = F G+ = G	A	A	B, C, D, E, F, G	A, AB, AC, AD, AE, AF, AG, ABC, ABD, ABE, ABF, ABG, ACD, ACE, ACF, ACG, ADE, ADF, ADG, AEF, AEG, AFG, ABCD, ABCE, ABCF, ABCG, ABDE, ABDF, ABDG, ABEF, ABEG, ABFG, ACDE, ACDF, ACDG, ACEF, ACEG, ACFG, ADEF, ADEG, ADFG, AEFG, ABCDE, ABCDF, ABCDG, ABCEF, ABCEG, ABCFG, ABDEF, ABDEG, ABDFG, ABEFG, ACDEF, ACDEG, ACDFG, ACEFG, ADEFG, ABCDEF, ABCDEG, ABCDFG, ABCEFG, ABDEFG, ACDEFG, ABCDEFG	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.

Auto Zone Tables Normalization

Tables	Attributes=short Forms	FD's	Closure	Candidate Key	Prime	non-prime	super keys	Highest Normal Form	Justification
bill	id = A (primary key) date = B mode_of_payment = C insurance_id = D customer_id= E job_id = F employee_id = G sale_type = H payment_plan_id = I	(A -> B, A->C, A->D, A->E, A->F, A->G, A->H, A->I)	A+ = ABCDEFGHI B+ = B C+ = C D+ = D E+ = E F+ = F G+ = G H+ = H I+ = I	A	A	B, C, D, E, F, G, H, I	A, AB, AC, AD, AE, AF, AG, AH, AI, ABC, ABD, ABE, ABF, ABG, ABH, ABI, ACD, ACE, ACF, ACG, ACH, ACI, ADE, ADF, ADG, ADH, ADI, AEF, AEG, AEH, AEI, AFG, AFH, AFI, AGH, AGI, AHI, ABCD, ABCE, ABCF, ABCG, ABCH, ABCI, ABDE, ABDF, ABDG, ABDH, ABDI, ABEF, ABEG, ABEH, ABEI..... so on (256 super keys)	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
job	id = A (Primary Key) job_date = B description = C customer_id = D automotive_retailer_id = E vin_number = F automobile_id = G job_status = H	(A -> B, A -> C, A -> F, A -> E, A -> H, F -> D , F -> G)	A+ = ABCDEFGH B+ = B C+ = C D+ = D E+ = E F+ = FDG G+ = G H+ = H	A	A	BCDEFGH	A,AB,AC,AD,AE,AF,AG,AH,ABC,ABD,ABE,ABF,ABG,ABH,ACDE,ACDF,ACDG,ACDH,ACEFG,ACEFH , so on..	The Table is is 2NF	*Not BCNF - The left-hand side of each FD in FD set is not a super key * Not 3NF - The right-hand of each FD in FD set is not a prime attribute. * It is 2NF - Every non-prime attribute is fully dependent on candidate key
part_service	id = A (Primary Key) name = B quantity = C job_id = D inventory_product_id = E type = F	(A -> B, A -> C, A -> D, A -> E, A -> F)	A+ = ABCDEF B+ = B C+ = C D+ = D E+ = E F+ = F	A	A	B, C, D, E, F	A, AB, AC, AD, AE, AF, ABC, ABD, ABE, ABF, ACD, ACE, ACF, ADE, ADF, AEF, ABCD, ABCE, ABCF, ACDE, ACDF, ACEF, ADEF, ABCDE, ABCDF, ABCEF, ACDEF, ABCDEF	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
payment_plan	id = A (Primary key) name = B installments = C interest = D	(A -> B, A->C, A->D)	A+ = ABCD B+ = B C+ = C D+ = D	A	A	B, C, D	A, AB, AC, AD, ABC, ABD, ACD, ABCD	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
customer	id = A (Primary Key) first_name = B last_name = C dob = D driverlicense = E phone = F address_id = G	(A -> F, A ->E, A -> G, E -> B, E ->C, E -> D)	A+ = AFEGBCD B+ = B C+ = C D+ = D E+ = EBCD F+ = F G+ G	A	A	BCDEFG	A,AB,AC,AD,AE,AF,ABC,ABD,ABE,ABF,ACDE,ACDF,ACDG,ACEF,ADEFG, so on..	The Table is is 2NF	*Not BCNF - The left-hand side of each FD in FD set is not a super key * Not 3NF - The right-hand of each FD in FD set is not a prime attribute. * It is 2NF - Every non-prime attribute is fully dependent on candidate key
insurance	id = A (Primary Key) policy_type = B provider = C claim_percentage = D	(A -> B, A -> C, A -> D)	A+ = ABCD B+ = B C+ = C D+ = D	A	A	B, C, D	A, AB, AC, AD, ABC, ABD, ACD, ABCD	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
inventory_product_supplier	inventory_product_id = A (Primary Key) supplier_id = B (Primary Key) quantity = C	AB -> C	A+ = A B+ = B C+ = C AB+ = ABC	AB	AB	C	AB, ABC	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
bill_inventory_product	bill_id = A (Primary Key) inventory_product_id = B (Primary Key) quantity = C	AB -> C	A+ = A B+ = B C+ = C AB+ = ABC	AB	AB	C	AB, ABC	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
	job_id = A (Primary Key)	As the table contains only primary keys there	A+ = A						All the Left Hand keys of FD's are

Auto Zone Tables Normalization

Tables	Attributes=short Forms	FD's	Closure	Candidate Key	Prime	non-prime	super keys	Highest Normal Form	Justification
job_employee	employee_id = B (Primary Key)	As the table contains only primary keys there will be no functional dependencies for this table.	B+ = B	AB	AB	{∅}	AB	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
			AB+ = AB						
customer_insurance	customer_id = A (Primary Key)	AB -> C	A+ = A	AB	AB	C	AB, ABC	BCNF	All the Left Hand keys of FD's are in superkeys, so the table is in BCNF.
	insurance_id = B (Primary Key)		B+ = B						
	status = C		C+ = C						
			AB+ = ABC						

Automotive Retailer Table Decomposition

Tables	Attributes=short Forms	FD's	Closure	Candidate Key	Prime	non-prime	super keys	Highest Normal Form	Justification
	Before Decomposition								
automotive_retailer	id = A (primary Key)	(A -> B, A -> E, A -> F, A -> G, B -> H, B -> C, B -> D)	A+ = ABCDEFG	A	A	BCDEFGH	A,AB,AC,AD,AE,AF,AG,AH,ABC,ABD,ABE,ABF,ABG,ABH,ACDE,ACDG,ADEFG,ABCDEFG, ABCDEFGH, so on..	The Table is is 2NF	*Not BCNF - The left-hand side of each FD in FD set is not a super key * Not 3NF - The right-hand of each FD in FD set is not a prime attribute. * It is 2NF - Every non-prime attribute is fully dependent on candidate key
	phone = B		B+ = BCDH						
	email = C		C+ = C						
	website = D		D+ = D						
	business_hours = E		E+ = E						
	manager_id = F		F+ = F						
	address_id = G		G+ = G						
	name = H		H+ = H						
	After Decomposition								
	automotive_retailer_contact table is decomposed from automotive_retailer								
automotive_retailer	id = A (primary Key)	(A -> B, A-> C, A -> D, A -> E)	A+ = ABCDE	A	A	BCDE	A,AB,AC,AD,ABC,ABD,ACD,ABCD,ABCE,ACDE,ABCDE so on..	The Table is in BCNF	In BCNF - The left-hand side of each FD i.e., id in FD set is a super key
	phone = B		B+ = B						
	business_hours = C		C+ = C						
	manager_id = D		D+ = D						
	address_id = E		E+ = E						
automotive_retailer_contact	phone = A (primary Key)	(A -> B, A -> C, A -> D)	A+ = ABCD	A	A	BCD	A,AB,AC,AD,ABC,ABD,ACD,ABCD	The Table is in BCNF	In BCNF - The left-hand side of each FD i.e., phone in FD set is a super key
	name = B		B+ = B						
	email = C		C+ = C						
	website = D		D+ = D						

Customer Table Decomposition

Tables	Attributes=short Forms	FD's	Closure	Candidate Key	Prime	non-prime	super keys	Highest Normal Form	Justification
customer	id = A (Primary Key)	(A -> F, A ->E, A -> G, E -> B, E ->C, E -> D)	A+ = AFEGBCD	A	A	BCDEFG	A,AB,AC,AD,AE,AF,ABC,ABD,ABE, ABF,ACDE,ACDF,ACDG,ACEF,AD EFG, so on..	The Table is is 2NF	*Not BCNF - The left-hand side of each FD in FD set is not a super key * Not 3NF - The right-hand of each FD in FD set is not a prime attribute. * It is 2NF - Every non-prime attribute is fully dependent on candidate key
	first_name = B		B+ = B						
	last_name = C		C+ = C						
	dob = D		D+ = D						
	driverlicense = E		E+ = EBCD						
	phone = F		F+ = F						
	address_id = G		G+ G						
	After Decomposition								
	customer_driverlicense table is decomposed from customer								
Customer	id = A (Primary Key)	(A -> B, A -> C, A -> D)	A+ = ABCD	A	A	BCD	A,AB,AC,AD,ABC,ABD,ACD,ABCD	The Table is in BCNF	In BCNF - The left-hand side of each FD i.e., id in FD set is a super key
	driverlicense = B		B+ = B						
	phone = C		C+ = C						
	address_id = D		D+ = D						
customer_driverlicense	driverlicense = A(primary key)	(A -> B, A -> C, A -> D)	A+ = ABCD	A	A	BCD	A,AB,AC,AD,ABC,ABD,ACD,ABCD	The Table is in BCNF	In BCNF - The left-hand side of each FD i.e., driverlicense in FD set is a super key
	first_name = B		B+ = B						
	last_name = C		C+ = C						
	dob = D		D+ = D						

Employee Table Decomposition

Tables	Attributes=short Forms	FD's	Closure	Candidate Key	Prime	non-prime	super keys	Highest Normal Form	Justification
	Before Decomposition								
Employee	id = A (primary Key)	(A -> F, A -> G, A -> H, A -> I, A -> J, A -> K H -> E, H -> D, H -> B, H -> C)	A+ = AFGHIJK	A	A	B, C, D, E, F, G, H, I, J, K	A,AB,AC,AD,AE,AF,AG,AH ,AI,AJ,AK,ABC,ABD,ABE,A BF,ABG,ABH,ABI,ABJ,ABK so on..	The Table is is 2NF	*Not BCNF - The left-hand side of each FD in FD set is not a super key * Not 3NF - The right-hand of each FD in FD set is not a prime attribute. * It is 2NF - Every non-prime attribute is fully dependent on candidate key
	first_name = B		B+ = B						
	last_name = C		C+ = C						
	dob = D		D+ = D						
	phone = E		E+ = E						
	email = F		F+ = F						
	annual_salary = G		G+ = G						
	ssn = H		H+ =HEDBC						
	automotive_retailer_id = I		I+ = I						
	address_id = J		J+ = J						
	hire_date = K		K+ = K						
	After Decomposition								
	SSN' table is decomposed from Employee								
SSN	SSN = A	(A -> B, A -> C, A -> D, A -> E, A -> F)	A+ = ABCDEF	A	A	B, C, D, E, F	A,AB,AC,AD,AE,AF,ABC,A BD,ABE,ABF,ACD,ACE,AC F,ADE,ADF,AEF,ABCD,AB CE,ABCF,ABDE	The Table is in BCNF	In BCNF - The left-hand side of each FD i.e., SSN in FD set is a super key
	first_name = B		B+ = B						
	last_name = C		C+ = C						
	phone = D		D+ = D						
	dob = E		E+ = E						
	gender = F		F+ = F						
Employee	id = A (primary Key)	(A -> B, A -> C, A -> D, A -> E, A -> F, A -> G)	A+ = ABCDEFG	A	A	B, C, D, E, F, G	A,AB,AC,AD,AE,AF,AG,AB C,ABD,ABE,ABF,ABG,ACD ,ACE,ACF,ACG,ADE,ADF, ADG,AEF	The Table is in BCNF	In BCNF - The left-hand side of each FD i.e., id in FD set is a super key Note: Here attribute 'ssn' is also unique per record, as it is a sensitive information, we didn't included in the candidate keys.
	email = B		B+ = B						
	annual_salary = C		C+ = C						
	ssn = D		D+ = D						
	automotive_retailer_id = E		E+ = E						
	address_id = F		F+ = F						
	hire_date = G		G+ = G						

Job Table Decomposition									
Tables	**Attributes=short Forms**	**FD's**	**Closure**	**Candidate Key**	**Prime**	**non-prime**	**super keys**	**Highest Normal Form**	**Justification**
job	id = A (Primary Key)	(A -> B, A -> C, A -> F, A -> E, A -> H, F -> D , F -> G)	A+ = ABCDEFGH	A	A	BCDEFGH	A,AB,AC,AD,AE,AF,AG,AH,ABC,ABD,ABE,ABF,ABG,ABH,ACDE,ACDF,ACDG,ACDH,ACEFG,ACEFH, so on..	The Table is is 2NF	*Not BCNF - The left-hand side of each FD in FD set is not a super key * Not 3NF - The right-hand of each FD in FD set is not a prime attribute. * It is 2NF - Every non-prime attribute is fully dependent on candidate key
	job_date = B		B+ = B						
	description = C		C+ = C						
	customer_id = D		D+ = D						
	automotive_retailer_id = E		E+ = E						
	vin_number = F		F+ = FDG						
	automobile_id = G		G+ = G						
	job_status = H		H+ = H						
	After Decomposition								
	Job_Customer table is decomposed from Job								
job	id = A (Primary Key)	(A -> B, A -> C, A -> D, A -> E, A -> F)	A+ = ABCDEF	A	A	BCDEF	A, AB, AC, AD, AE, AF, ABC, ABD, ABE, ABF, ACD, ACE, ACF, ADE, ADF, AEF, ABCD, ABCE, ABCF, ABDE, son on...	The Table is in BCNF	In BCNF - The left-hand side of each FD i.e., id in FD set is a super key
	job_date = B		B+ = B						
	description = C		C+ = C						
	vin_number = D		D+ = D						
	automotive_retailer_id = E		E+ = E						
	job_status = F		F+ = F						
job_customer	vin_number = A	(A -> B, A -> C)	A+ = ABC	A	A	BC	A,AB,AC,ABC	The Table is in BCNF	In BCNF - The left-hand side of each FD i.e., vin_number in FD set is a super key
	customer_id = B		B+ = B						
	automobile_id = C		C+ = C						

Employee Table Decomposition Implementation:

Now, let's discuss how we have implemented LLJD-BCNF Algorithm to decompose Employee table. Please find the implementation of the algorithm below table.

Employee table:

The below are the attributes and their respective short forms of the table **Employee**.

- id = A (primary Key)
- first_name = B
- last_name = C
- dob = D
- phone = E
- email = F
- annual_salary = G
- ssn = H
- automotive_retailer_id = I
- address_id = J
- hire_date = K

R (A, B, C, D, E, F, G, H, I, J, K)
FD set is (A -> F, A -> G, A -> H, A -> I, A -> J, A -> K, H -> E, H -> D, H -> B, H -> C)

Please find the closures of each attribute below:

- A+ = AFGHIJK
- B+ = B
- C+ = C
- D+ = D
- E+ = E
- F+ = F
- G+ = G
- H+ =HEDBC
- I+ = I
- J+ = J
- K+ = K

Candidate Key: A

Current Highest NF: 2NF:

- Not BCNF - The left-hand side of each FD in FD set is not a super key.
- Not 3NF - The right-hand of each FD in FD set is not a prime attribute.
- It is 2NF - Every non-prime attribute is fully dependent on candidate key.

We need to decompose the given table 'R' such that decomposed tables are in BCNF. Now, let's start with decomposition using LLJD-BCNF algorithm.

LLJD-BCNF algorithm:

- Initialization: D = {}
- 1st iteration: R (A, B, C, D, E, F, G, H, I, J, K) is not in BCNF because H+ =HEDBC is a not trivial FD and H is not a super key. R (A, B, C, D, E, F, G, H, I, J, K) can be divided into R1(A, F, G, H, I, J, K) and R2(H, B, C, D, E) as A is deriving { F, G, H, I, J K } and H is deriving as { B, C, D, E}.
- R1 has FD set F1 = {A -> F, A -> G, A -> H, A -> I, A -> J, A -> K}, candidate key is A.
- R2 has FD set F2 = {H -> B, H ->C, H ->D, H -> E}, candidate key is H.
- Check the decomposed tables are in BCNF or not –
    - R1 is in BCNF because in AFGHIJK, A is a super key.
    - R2 is in BCNF because in HBCDE, H is a super key.
- Now the algorithm stops here, as the decomposed tables are in BCNF.
- Finally, D = {R1(A, F, G, H, I, J, K), R2(H, B, C, D, E)}

R (A, B, C, D, E, F, G, H, I, J, K) is decomposed into R1(A, F, G, H, I, J, K) and R2(H, B, C, D, E).

Now, we have named R2 table as "SSN" and R1 table will be Employee. All the respective columns of Employee table are migrated to SSN table and dropped redundant columns in Employee table.

Proof: Decomposition is loss less:

To find given decomposition is loss less or not, first we need to find the "R1 ∩ R2", then result of "R1 – R2" or "R2 – R1" should be derived by the result of "R1 ∩ R2".

In this case
• "R1∩R2" = H
• "R1–R2" = A, F, G, I, J, K
• "R2–R1" = B, C, D, E

H+ = HBCDE

'H' can derive 'B C D E' (R2 – R1). So, the decomposition {R1, R2} is a loss less join.

Proof: Decomposition is dependency preserving:

A -> F is preserved in R1
A -> G is preserved in R1
A -> H is preserved in R1
A -> I is preserved in R1
A -> J is preserved in R1
A -> K is preserved in R1
H -> E is preserved in R2
H -> D is preserved in R2
H -> B is preserved in R2
H -> C is preserved in R2

From all the above, we can derive that all the functional dependencies are preserved in the decomposition.

Automotive Retailer Table Decomposition Implementation:

Now, let's discuss how we have implemented LLJD-BCNF Algorithm to decompose automotive_retailer table. Please find the implementation of the algorithm below table.

Automotive Retailer table:

The below are the attributes and their respective short forms of the table **Automotive Retailer**.

- id = A (primary Key)
- phone = B
- email = C
- website = D
- business_hours = E
- manager_id = F
- address_id = G
- name = H

R (A, B, C, D, E, F, G, H)
FD set is (A -> B, A -> E, A -> F, A -> G, B -> H, B -> C, B -> D)

Please find the closures of each attribute below:

- A+ = ABCDEFGH
- B+ = BCDH
- C+ = C

- D+ = D
- E+ = E
- F+ = F
- G+ = G
- H+ = H

Candidate Key: A

Current Highest NF: 2NF:

- Not BCNF - The left-hand side of each FD in FD set is not a super key.
- Not 3NF - The right-hand of each FD in FD set is not a prime attribute.
- It is 2NF - Every non-prime attribute is fully dependent on candidate key.

We need to decompose the given table 'R' such that decomposed tables are in BCNF. Now, let's start with decomposition using LLJD-BCNF algorithm.

LLJD-BCNF algorithm:

- Initialization: D = {}
- 1st iteration: R (A, B, C, D, E, F, G, H) is not in BCNF because B+ = BCDH is a not trivial FD and B is not a super key. R (A, B, C, D, E, F, G, H) can be divided into R1(A, B, E, F, G) and R2(B, C, D, H) as A is deriving {B, E, F, G} and B is deriving as {H, C, D}.
- R1 has FD set F1 = {A -> B, A -> E, A -> F, A -> G}, candidate key is A.
- R2 has FD set F2 = {B -> H, B -> C, B -> D}, candidate key is B.
- Check the decomposed tables are in BCNF or not –
    - R1 is in BCNF because in ABEFG, A is a super key.
    - R2 is in BCNF because in BCDH, B is a super key.
- Now the algorithm stops here, as the decomposed tables are in BCNF.
- Finally, D = {R1(A, B, E, F, G), R2(B, C, D, H)}

R (A, B, C, D, E, F, G, H) is decomposed into R1(A, B, E, F, G) and R2(B, C, D, H).

Now, we have named R2 table as "automotive_retailer_contact" and R1 table will be automotive_retailer. All the respective columns of automotive_retailer table are migrated to automotive_retailer_contact table and dropped redundant columns in automotive_retailer table.

Proof: Decomposition is loss less:

To find given decomposition is loss less or not, first we need to find the "R1 ∩ R2", then result of "R1 – R2" or "R2 – R1" should be derived by the result of "R1 ∩ R2".

In this case
• "R1∩R2" = B

• “R1–R2” = A, E, F, G
• “R2–R1” = C, D, H

B+ = BCDH

‘B’ can derive ‘B C D H’ (R2 – R1). So, the decomposition {R1, R2} is a loss less join.

Proof: Decomposition is dependency preserving:

A -> B is preserved in R1
A -> E is preserved in R1
A -> F is preserved in R1
A -> G is preserved in R1
B -> H is preserved in R2
B -> C is preserved in R2
B -> D is preserved in R2

From all the above, we can derive that all the functional dependencies are preserved in the decomposition.

Customer Table Decomposition Implementation:

Now, let’s discuss how we have implemented LLJD-BCNF Algorithm to decompose Customer table. Please find the implementation of the algorithm below table.

Customer table:

The below are the attributes and their respective short forms of the table **Customer**.

- id = A (Primary Key)
- first_name = B
- last_name = C
- dob = D
- driverlicense = E
- phone = F
- address_id = G

R (A, B, C, D, E, F, G)
FD set is "(A -> F, A ->E, A -> G, E -> B, E ->C, E -> D)"

Please find the closures of each attribute below:

- A+ = AFEGBCD
- B+ = B
- C+ = C
- D+ = D
- E+ = EBCD
- F+ = F
- G+ G

Candidate Key: A

Current Highest NF: 2NF:

- Not BCNF - The left-hand side of each FD in FD set is not a super key.
- Not 3NF - The right-hand of each FD in FD set is not a prime attribute.
- It is 2NF - Every non-prime attribute is fully dependent on candidate key.

We need to decompose the given table 'R' such that decomposed tables are in BCNF. Now, let's start with decomposition using LLJD-BCNF algorithm.

LLJD-BCNF algorithm:

- Initialization: D = {}
- 1st iteration: R (A, B, C, D, E, F, G) is not in BCNF because E+ = EBCD is a not trivial FD and E is not a super key. R (A, B, C, D, E, F, G) can be divided into R1(A, F, E, G) and R2(E, B, C, D) as A is deriving {F, E, G} and E is deriving as {B, C, D}.
- R1 has FD set F1 = {A -> F, A ->E, A -> G}, candidate key is A.
- R2 has FD set F2 = {E -> B, E ->C, E -> D}, candidate key is E.
- Check the decomposed tables are in BCNF or not –
    - R1 is in BCNF because in AFEG, A is a super key.
    - R2 is in BCNF because in EBCD, E is a super key.
- Now the algorithm stops here, as the decomposed tables are in BCNF.
- Finally, D = {R1(A, F, E, G), R2(E, B, C, D)}

R (A, B, C, D, E, F, G) is decomposed into R1(A, F, E, G) and R2(E, B, C, D).

Now, we have named R2 table as "customer_driverlicense" and R1 table will be" customer". All the respective columns of Customer table are migrated to customer_driverlicense table and dropped redundant columns in Customer table.

Proof: Decomposition is loss less:

To find given decomposition is loss less or not, first we need to find the "R1 ∩ R2", then result of "R1 – R2" or "R2 – R1" should be derived by the result of "R1 ∩ R2".

In this case
• "R1∩R2" = E
• "R1–R2" = A, F, G
• "R2–R1" = B, C, D

E+ = EBCD

'E' can derive 'E B C D' (R2 – R1). So, the decomposition {R1, R2} is a loss less join.

Proof: Decomposition is dependency preserving:

A -> F is preserved in R1
A ->E is preserved in R1
A -> G is preserved in R1
E -> B is preserved in R2
E ->C is preserved in R2
E -> D is preserved in R2

From all the above, we can derive that all the functional dependencies are preserved in the decomposition.

Job Table Decomposition Implementation:

Now, let's discuss how we have implemented LLJD-BCNF Algorithm to decompose Job table. Please find the implementation of the algorithm below table.

Job table:

The below are the attributes and their respective short forms of the table **Job**.

- id = A (Primary Key)
- job_date = B
- description = C
- customer_id = D
- automotive_retailer_id = E
- vin_number = F
- automobile_id = G

- job_status = H

R (A, B, C, D, E, F, G, H)
FD set is (A -> B, A -> C, A -> F, A -> E, A -> H, F -> D, F -> G)

Please find the closures of each attribute below:

- A+ = ABCDEFGH
- B+ = B
- C+ = C
- D+ = D
- E+ = E
- F+ = FDG
- G+ = G
- H+ = H

Candidate Key: A

Current Highest NF: 2NF:

- Not BCNF - The left-hand side of each FD in FD set is not a super key.
- Not 3NF - The right-hand of each FD in FD set is not a prime attribute.
- It is 2NF - Every non-prime attribute is fully dependent on candidate key.

We need to decompose the given table 'R' such that decomposed tables are in BCNF. Now, let's start with decomposition using LLJD-BCNF algorithm.

LLJD-BCNF algorithm:

- Initialization: D = {}
- 1st iteration: R (A, B, C, D, E, F, G, H) is not in BCNF because F+ = FDG is a non-trivial FD and F is not a super key. R (A, B, C, D, E, F, G, H) can be divided into R1(A, B, C, F, E, H) and R2(F, D, G) as A is deriving {B, C, F, E, H} and F is deriving as {D, G}.
- R1 has FD set F1 = {A -> B, A -> C, A -> F, A -> E, A -> H}, candidate key is A.
- R2 has FD set F2 = {F -> D, F -> G}, candidate key is F.
- Check the decomposed tables are in BCNF or not –
    - R1 is in BCNF because in ABCFEH, A is a super key.
    - R2 is in BCNF because in FDG, F is a super key.
- Now the algorithm stops here, as the decomposed tables are in BCNF.
- Finally, D = {R1(A, B, C, F, E, H), R2(F, D, G)}

R (A, B, C, D, E, F, G, H) is decomposed into R1(A, B, C, F, E, H) and R2(F, D, G).

Now, we have named R2 table as “job_customer” and R1 table will be job. All the respective columns of job table are migrated to job_customer table and dropped redundant columns in job table.

Proof: Decomposition is loss less:

To find given decomposition is loss less or not, first we need to find the “R1 ∩ R2”, then result of “R1 – R2” or “R2 – R1” should be derived by the result of “R1 ∩ R2”.

In this case
- “R1∩R2” = F
- “R1–R2” = A, B, C, E, H
- “R2–R1” = D, G

F+ = FDG

‘F’ can derive ‘F D G’ (R2 – R1). So, the decomposition {R1, R2} is a loss less join.

Proof: Decomposition is dependency preserving:

A -> B is preserved in R1
A -> C is preserved in R1
A -> F is preserved in R1
A -> E is preserved in R1
A -> H is preserved in R1
F -> D is preserved in R2
F -> G is preserved in R2

From all the above, we can derive that all the functional dependencies are preserved in the decomposition.

Implementation on Database:

Automotive Retailer:

Decomposition implementation of Automotive Retailer.

Before decomposition:

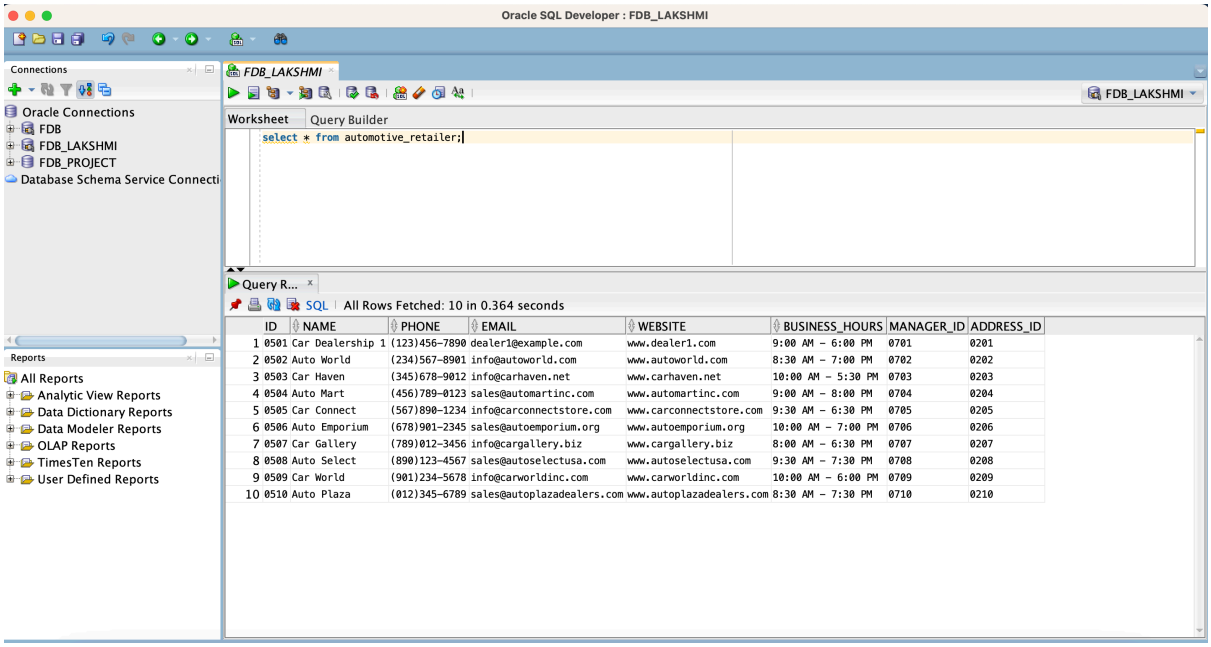


Figure-1: Displayed automotive_retailer table before Decomposition.

Explanation: The Query "**select * from automotive_retailer;**" is used to display the complete automotive_retailer table data.

After Decomposition:

Decomposition of automotive_retailer table into automotive_retailer_contact and automotive_retailer.

Creation of automotive_retailer_contact table:

```
CREATE TABLE automotive_retailer_contact (
   PHONE VARCHAR2(20) not null primary key,
   NAME VARCHAR2(30) not null,
   EMAIL VARCHAR (255) not null,
   WEBSITE VARCHAR (255) not null
);
```

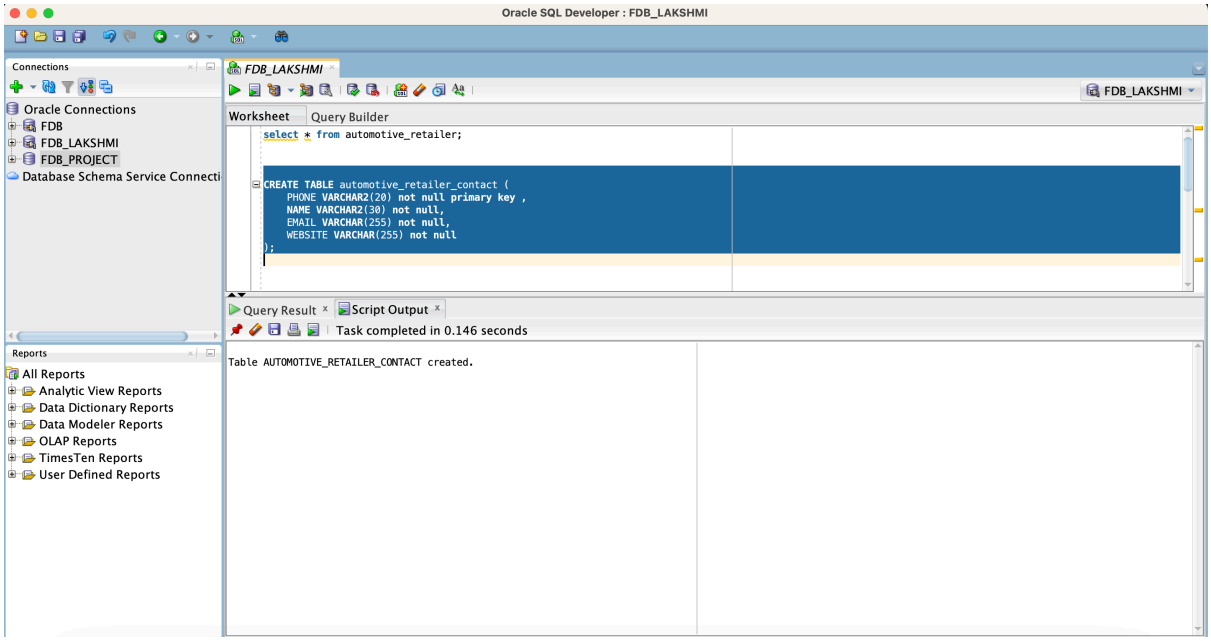


Figure-2: Created automotive_retailer_contact Table.

Explanation: This SQL statement creates a table named automotive_retailer_contact with four columns to store contact information for automotive retailers. The PHONE column is a primary key, ensuring each entry has a unique phone number. The NAME column stores the retailer's name, while the EMAIL column holds their email address. The WEBSITE column is intended for storing the retailer's website URL. All columns are defined as not null to enforce data integrity by requiring values for each contact information field.

Automotive_retailer_contact Data Migration:

"INSERT INTO automotive_retailer_contact (PHONE, NAME, EMAIL, WEBSITE) select PHONE, NAME, EMAIL, WEBSITE from automotive_retailer;"

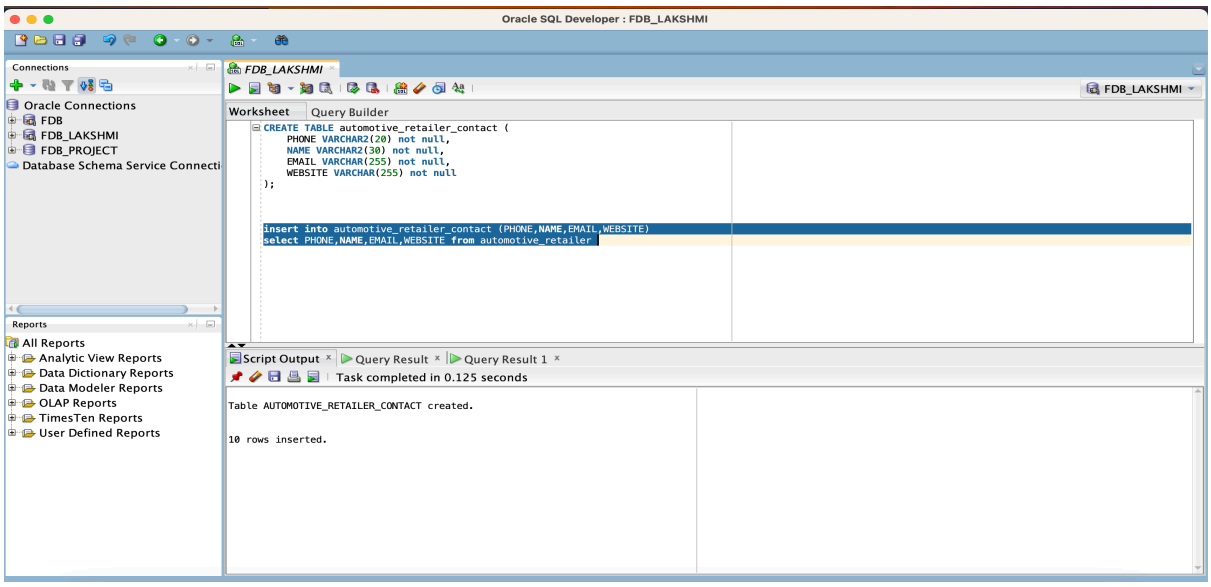


Figure-3: Migrated data to automotive_retailer_contact Table.

Explanation: As this automotive_retailer_contact table is decomposed from automotive_retailer table. We need to migrate the data from automotive_retailer table to automotive_retailer_contact table.

automotive_retailer_contact after Data Migration:

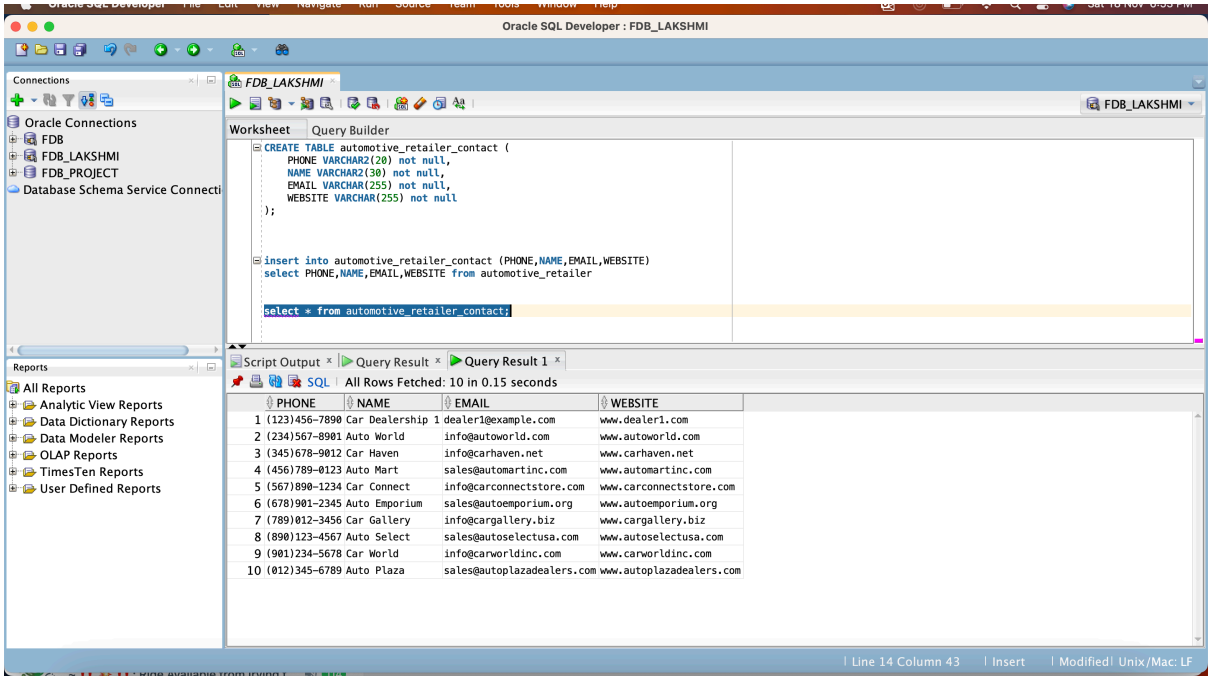


Figure-4: After migration.

Altering the automotive_retailer Table:

Now, after migration we will drop the redundant columns in automotive_retailer table.Below Screenshot represents the dropping of columns and making ‘phone’ column as the foreign key referencing the ‘phone’ column in automotive_retailer_contact table.

alter table automotive_retailer drop column NAME;
alter table automotive_retailer drop column EMAIL;
alter table automotive_retailer drop column WEBSITE;

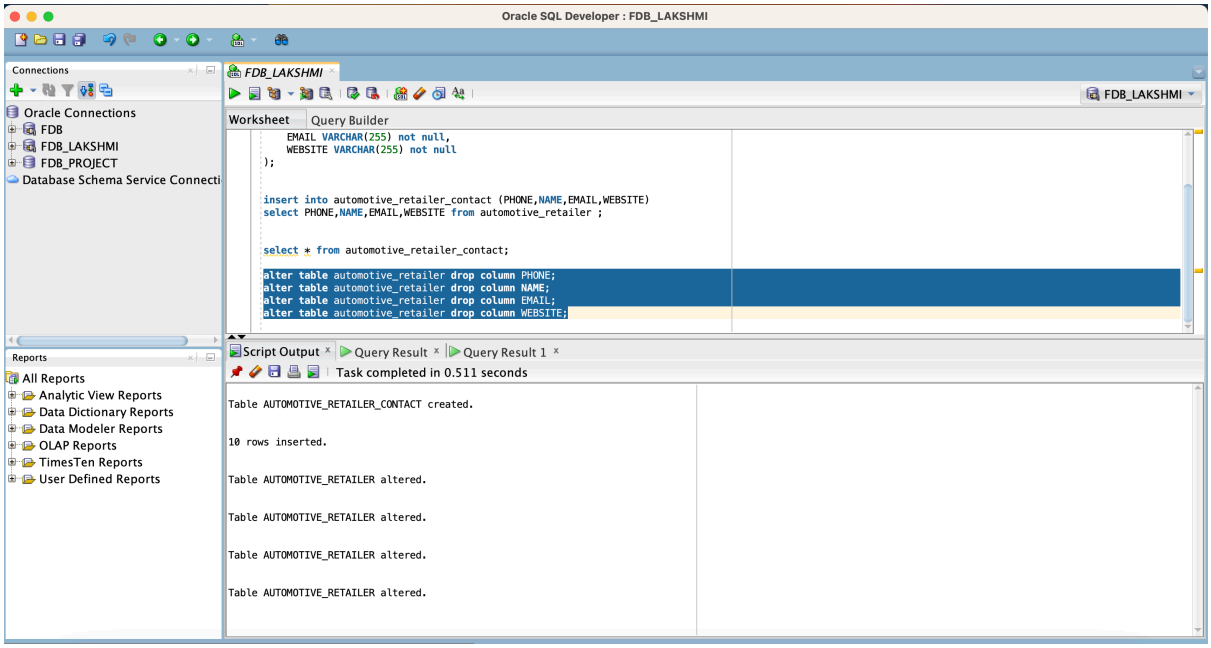


Figure-5: Altering the table.

ALTER TABLE automotive_retailer ADD CONSTRAINT fk_phone FOREIGN KEY (phone) REFERENCES automotive_retailer_contact(phone);

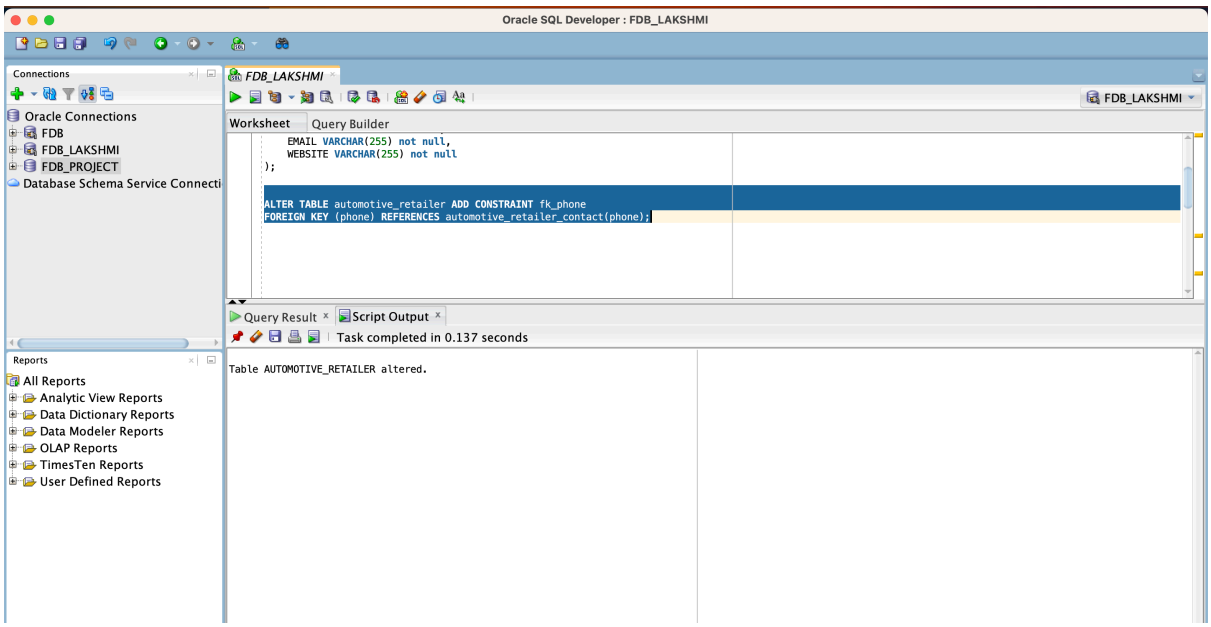


Figure-6: Altering the table.

Automotive Retailer table after migration:

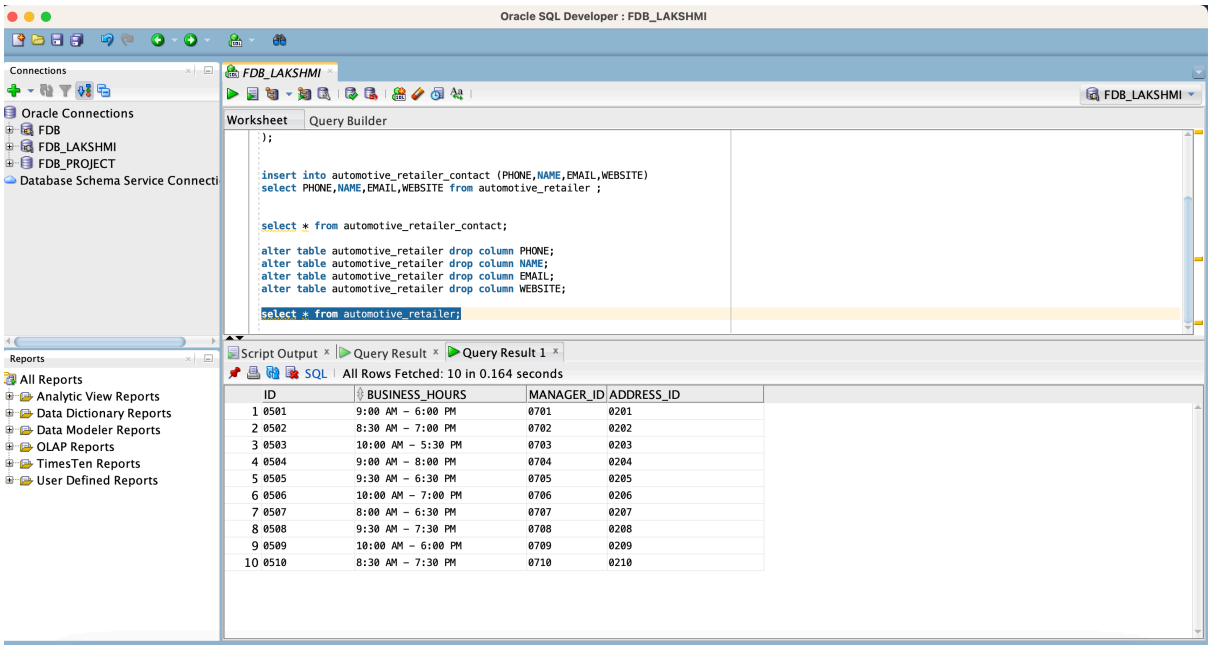


Figure-7: After migration and alteration.

Customer:

Decomposition implementation of Customer.

Before decomposition:

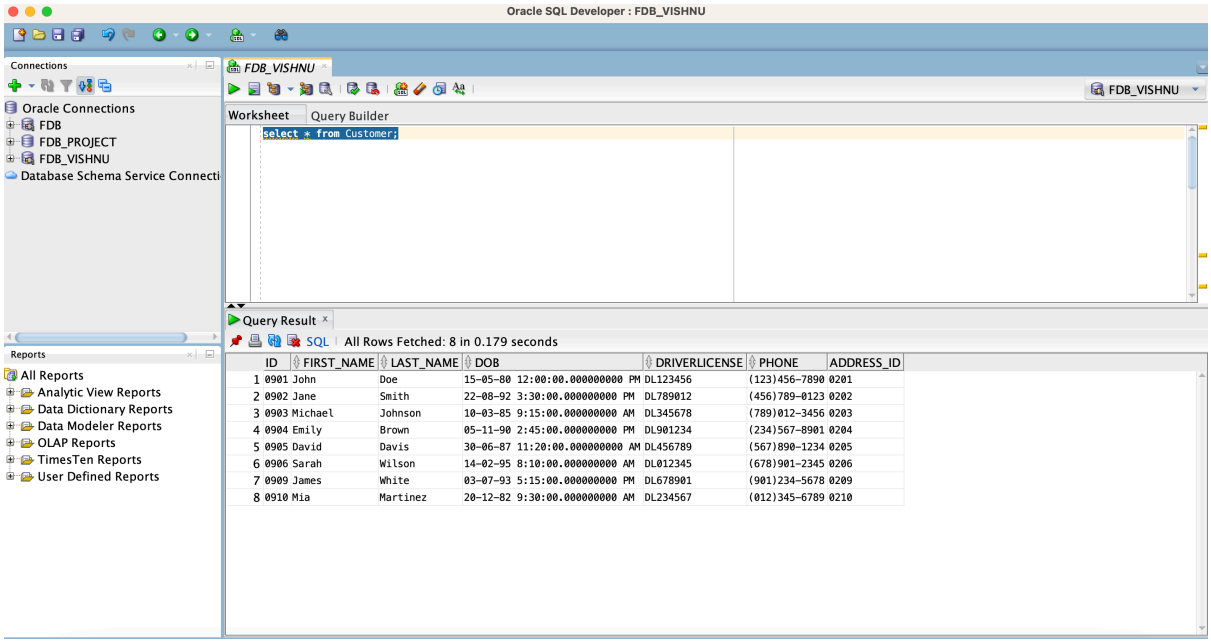


Figure-8: Before decomposition.

Explanation: The Query "**select * from customer;**" is used to display the complete customer table data.

After Decomposition:

Decomposition of Customer table into customer_driverlicense and customer.

Creation of customer_driverlicense table:

```
CREATE TABLE customer_driverlicense (
   first_name VARCHAR(100) not null,
   last_name VARCHAR(100) not null,
   dob TIMESTAMP(6),
   DRIVERLICENSE VARCHAR2(50) primary key
);
```

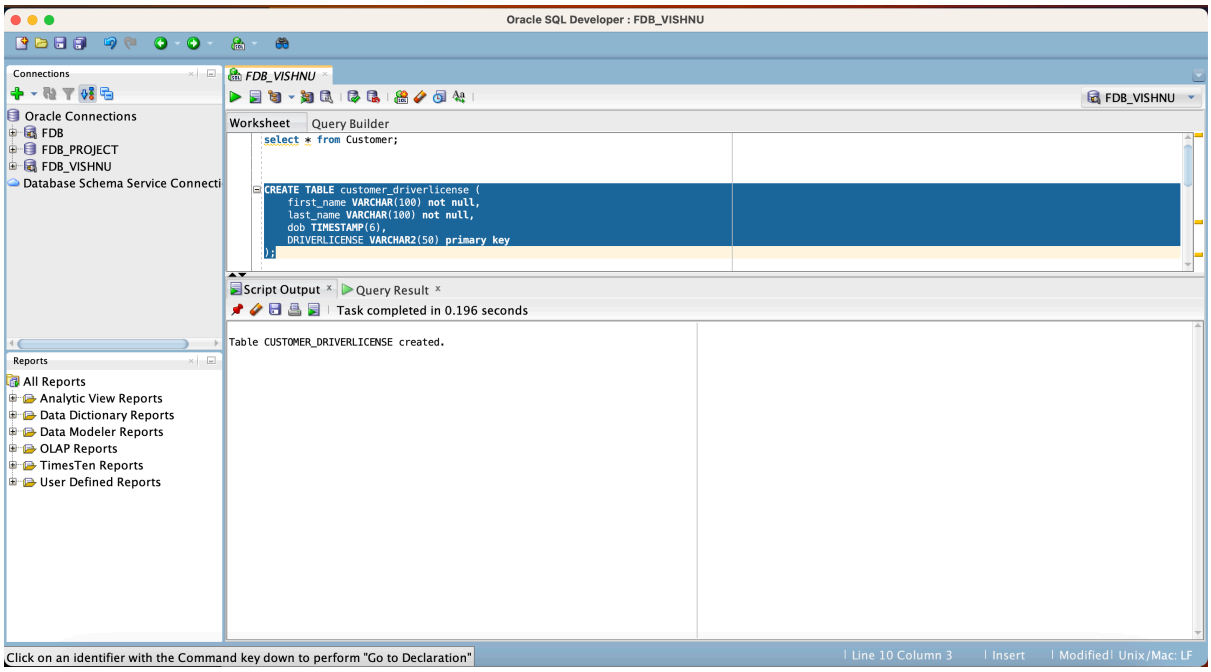


Figure-9: Creation of the new table.

Explanation: This SQL statement creates a table named customer_driverlicense to store information related to customer driver's licenses. The table has four columns: first_name and last_name to store the customer's first and last names, respectively. The dob column is of type TIMESTAMP with precision 6, representing the customer's date of birth. The DRIVERLICENSE column is the primary key, ensuring a unique identifier for each customer's driver's license within the table. All columns marked as not null indicate that these fields must have values to maintain data integrity.

customer_driverlicense table Data Migration:

"INSERT INTO customer_driverlicense (first_name,last_name,dob,DRIVERLICENSE) select first_name, last_name, dob, DRIVERLICENSE from customer;"

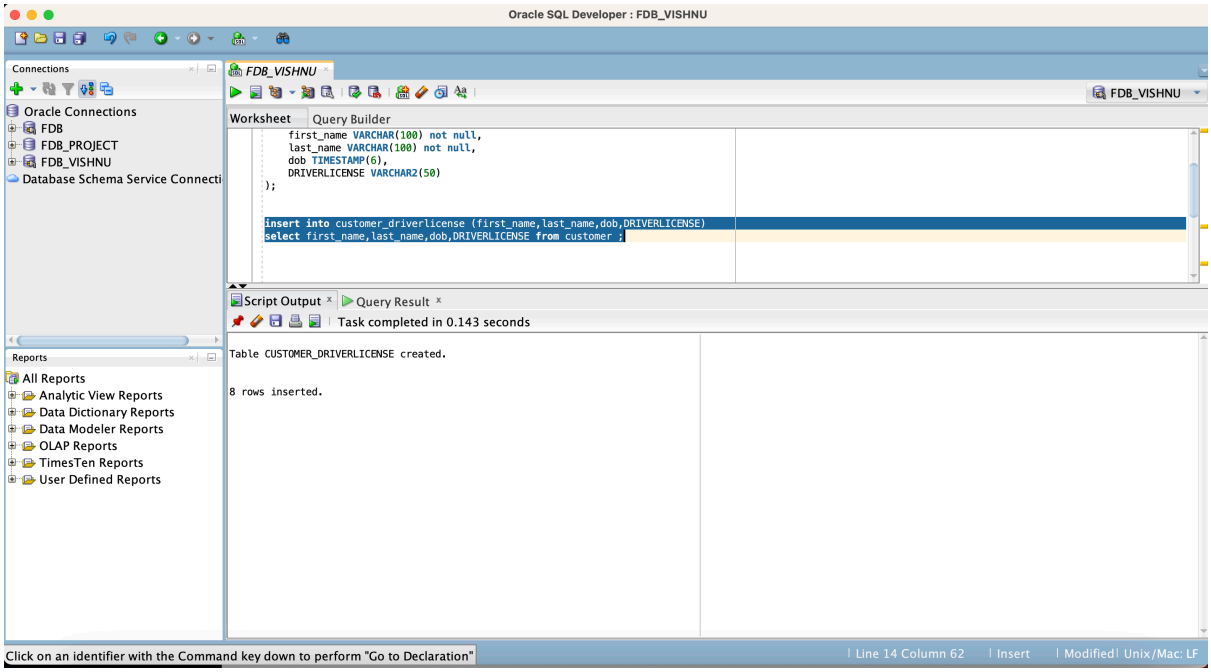


Figure-10: Data Migration.

Explanation: As this customer_driverlicense table is decomposed from Customer table. We need to migrate the data from Customer table to customer_driverlicense table.

customer_driverlicense after Data Migration:

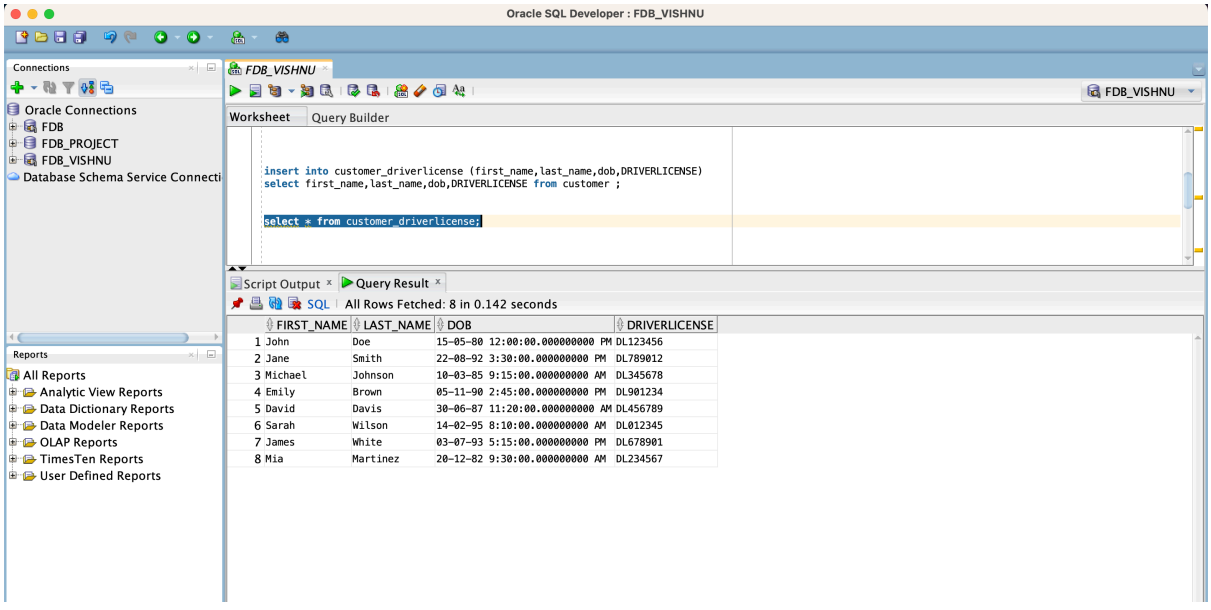


Figure-11: After Data Migration.

Altering the Customer Table:

Now, after migration we will drop the redundant columns in Customer table. Below Screenshot represents the dropping of columns and making 'driverlicense' column as the foreign key referencing the 'driverlicense' column in customer_driverlicense table.

```
alter table customer drop column first_name;
alter table customer drop column last_name;
alter table customer drop column dob;
ALTER TABLE customer ADD CONSTRAINT fk_customer_driverlicense
FOREIGN KEY (DRIVERLICENSE) REFERENCES customer(DRIVERLICENSE);
```

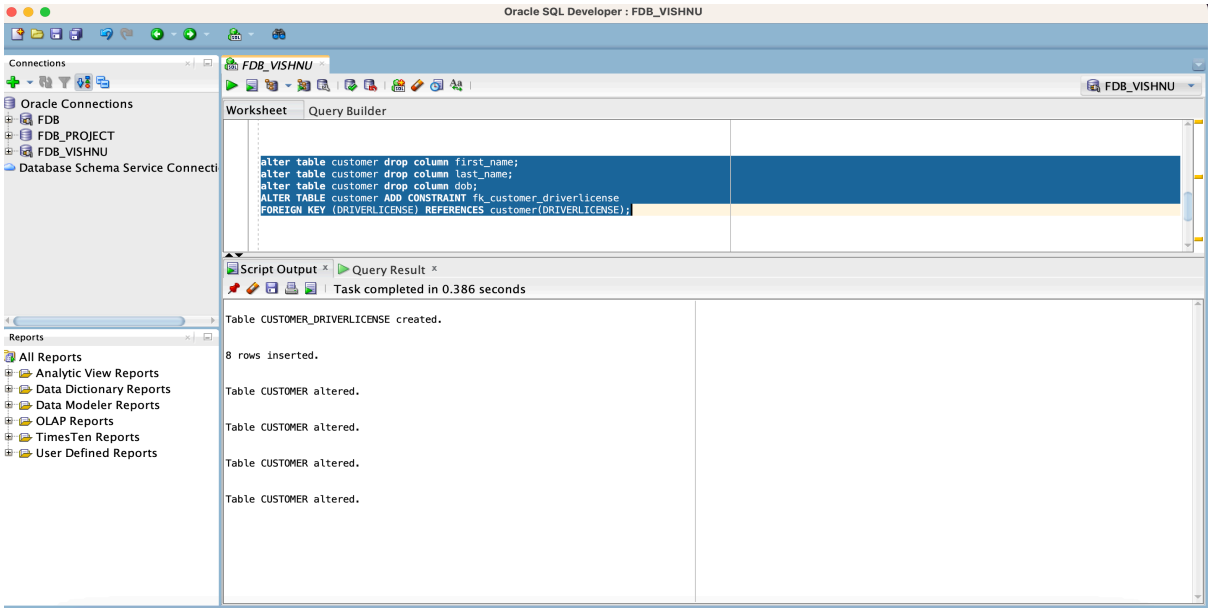


Figure-12: Altering the table.

Customer table after migration:

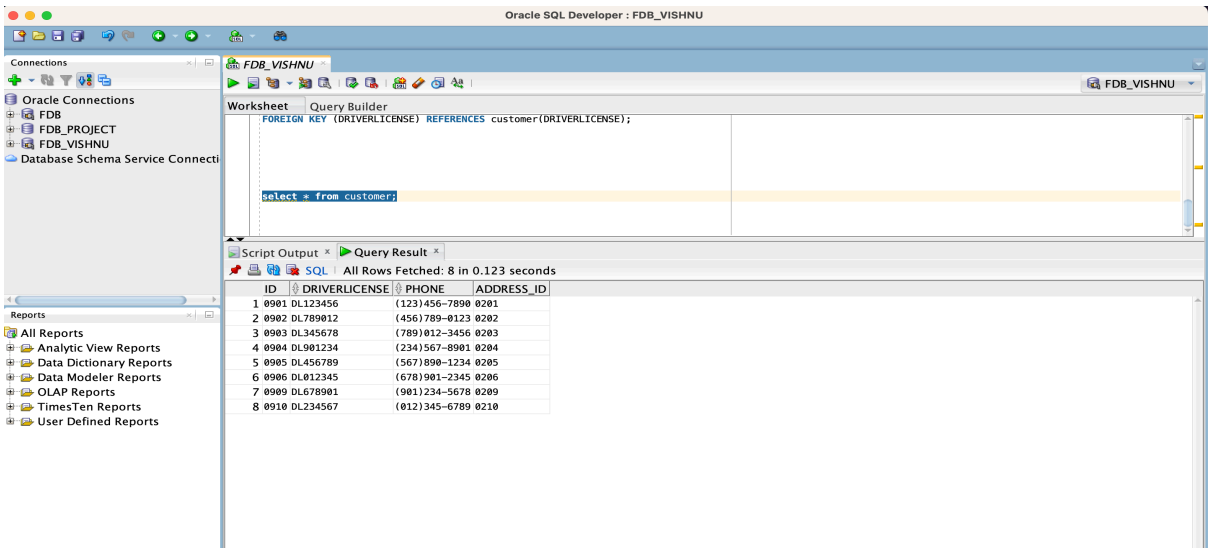


Figure-13: After Altering the table.

JOB:

Decomposition implementation on Job table.

Before Decomposition:

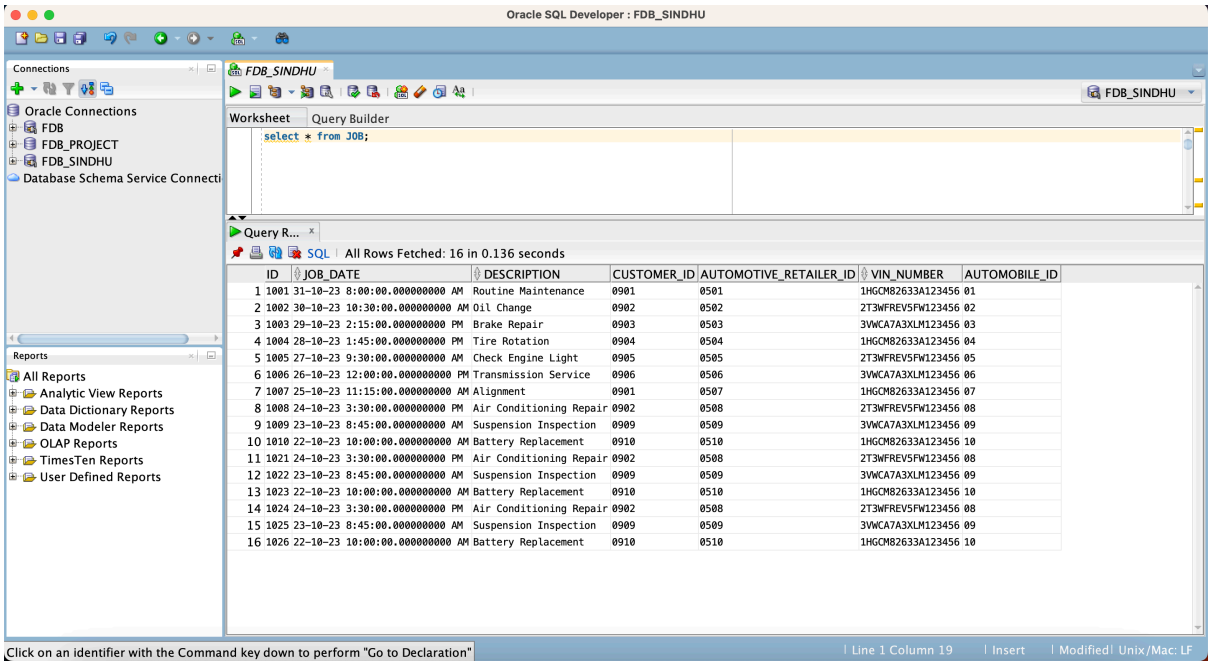


Figure-14: Before Decomposition.

Explanation: The Query **"select * from Job;"** is used to display the complete JOB table data.

After Decomposition:

Creation of Job_customer table:

```
CREATE TABLE Job_Customer (
   vin_number VARCHAR2(100) not null primary key,
   customer_id RAW(16) not null,
   automobile_id RAW(16) not null ,
   FOREIGN KEY (customer_id) REFERENCES customer(id),
   FOREIGN KEY (automobile_id) REFERENCES automobile(id)
);
```

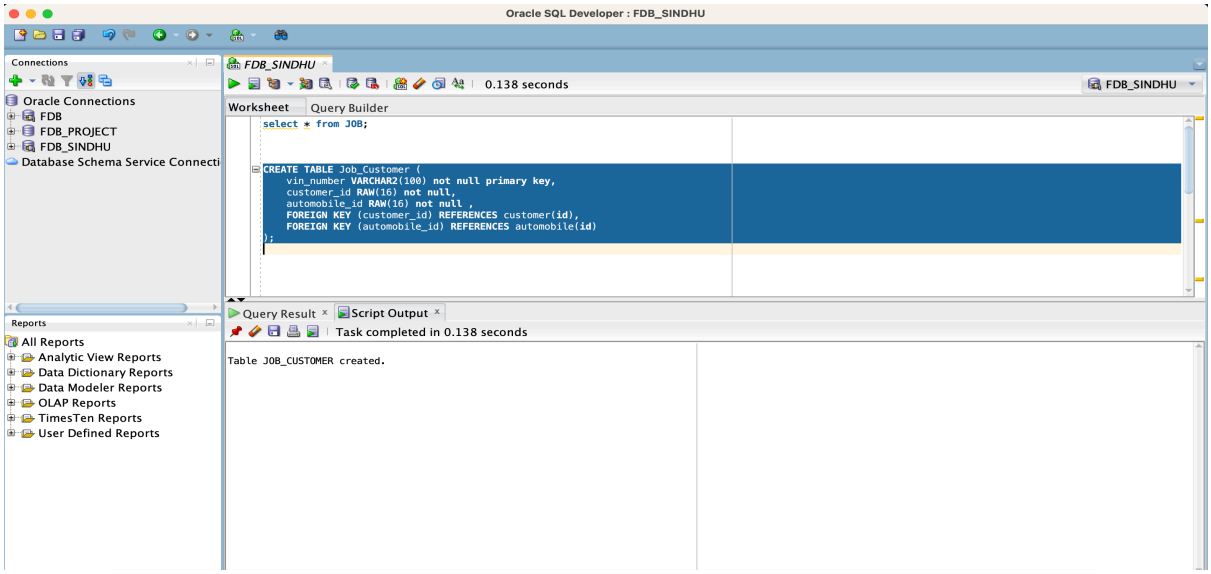


Figure-15: Table Creation.

Explanation: This SQL statement creates a table named Job_Customer to establish a relationship between jobs, customers, and automobiles. The vin_number column serves as the primary key for this table, representing a unique identifier for each job associated with a specific vehicle. The customer_id and automobile_id columns are foreign keys referencing the id column in the customer and automobile tables, respectively. These foreign key constraints enforce referential integrity, ensuring that each entry in the Job_Customer table corresponds to valid customer and automobile records. The RAW(16) data type suggests the use of 16-byte raw data for identifiers, which could be UUIDs or other unique identifiers. The not null constraint is applied to these columns to maintain data integrity by requiring valid references.

job_customer Data Migration:
"INSERT INTO Job_Customer (vin_number,customer_id,automobile_id)
select vin_number, customer_id, automobile_id from JOB;"

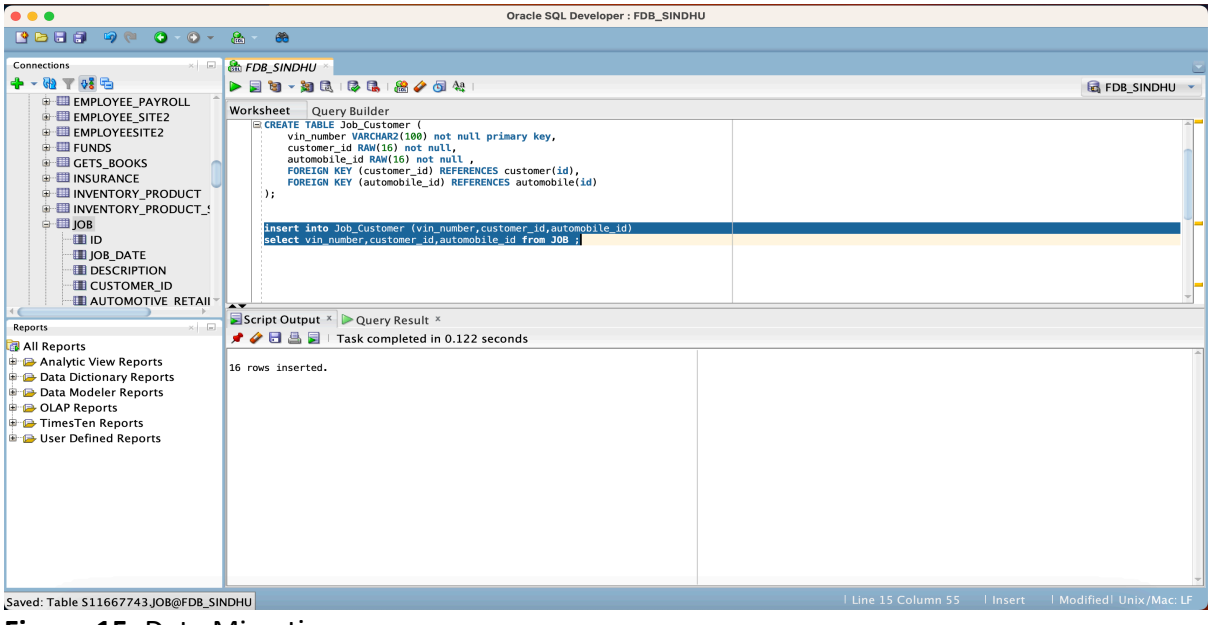


Figure-15: Data Migration.

Explanation: As this Job_Customer table is decomposed from Job table. We need to migrate the data from Job table to Job_Customer table.

Job_Customer after Data Migration:

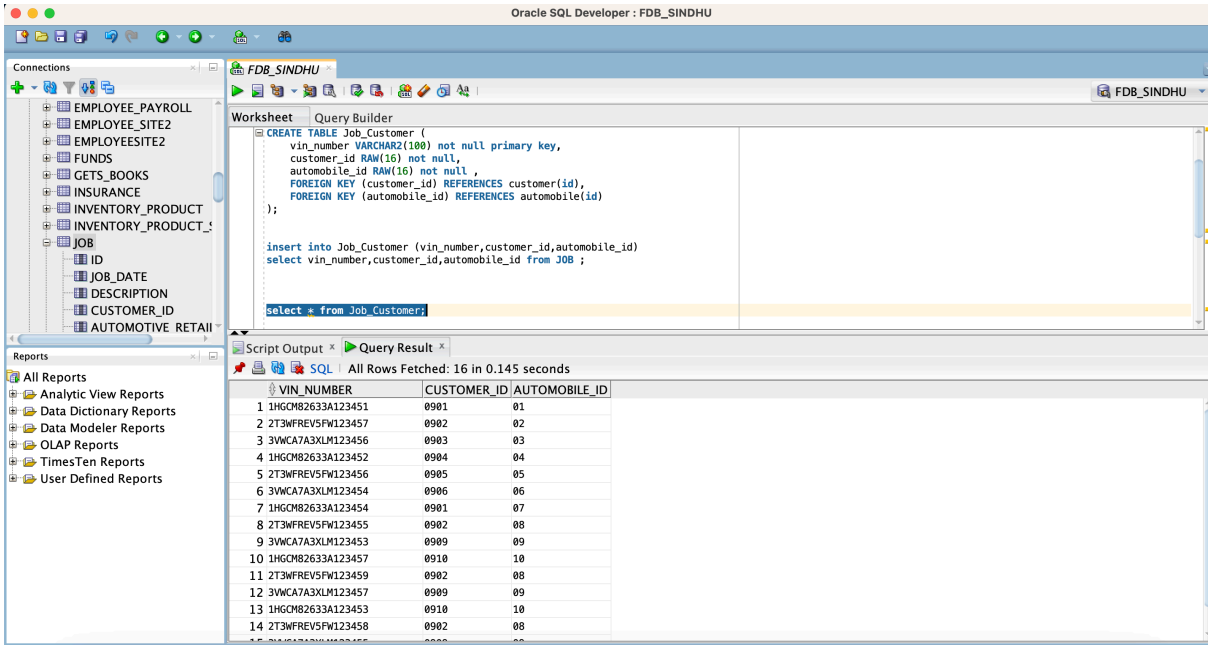


Figure-15: After Data Migration.

Altering the Job Table:

Now, after migration we will drop the redundant columns in Job table. Below Screenshot represents the dropping of columns and making ‘vin_number’ column as the foreign key referencing the ‘vin_number’ column in Job_Customer table.

alter table Job drop column customer_id;
alter table Job drop column automobile_id;
ALTER TABLE Job ADD CONSTRAINT fk_vin_number
FOREIGN KEY (vin_number) REFERENCES Job_Customer(vin_number);

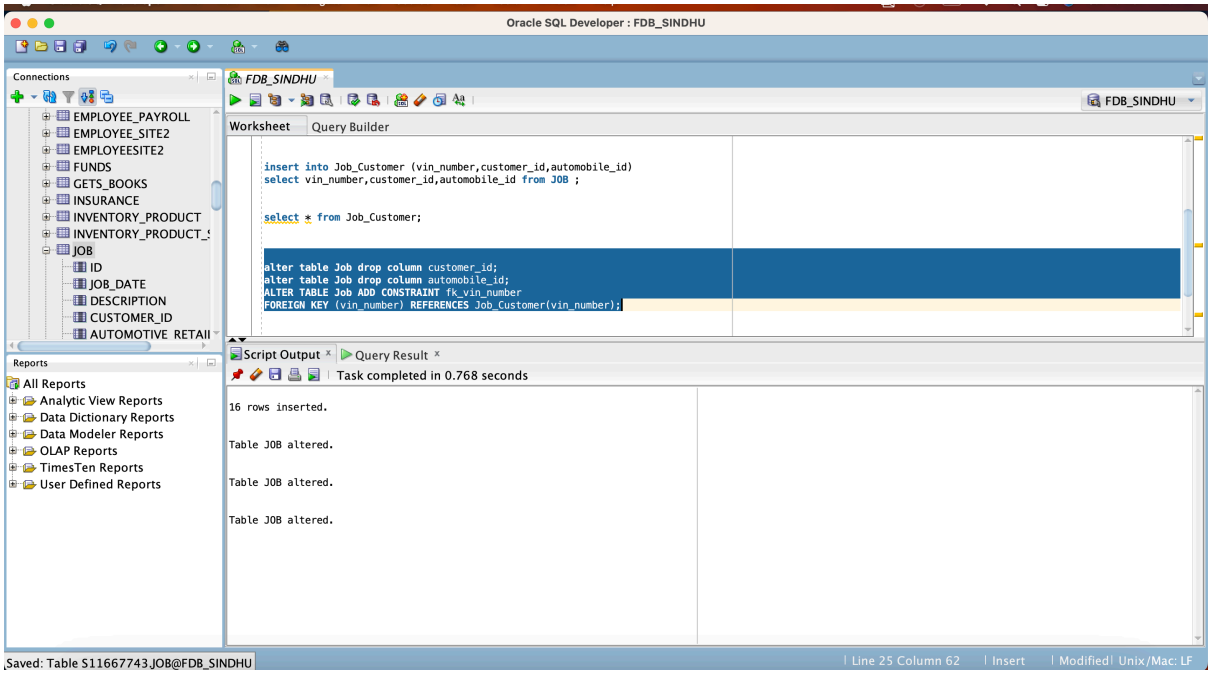


Figure-16: Altering the table.

Job table after migration:

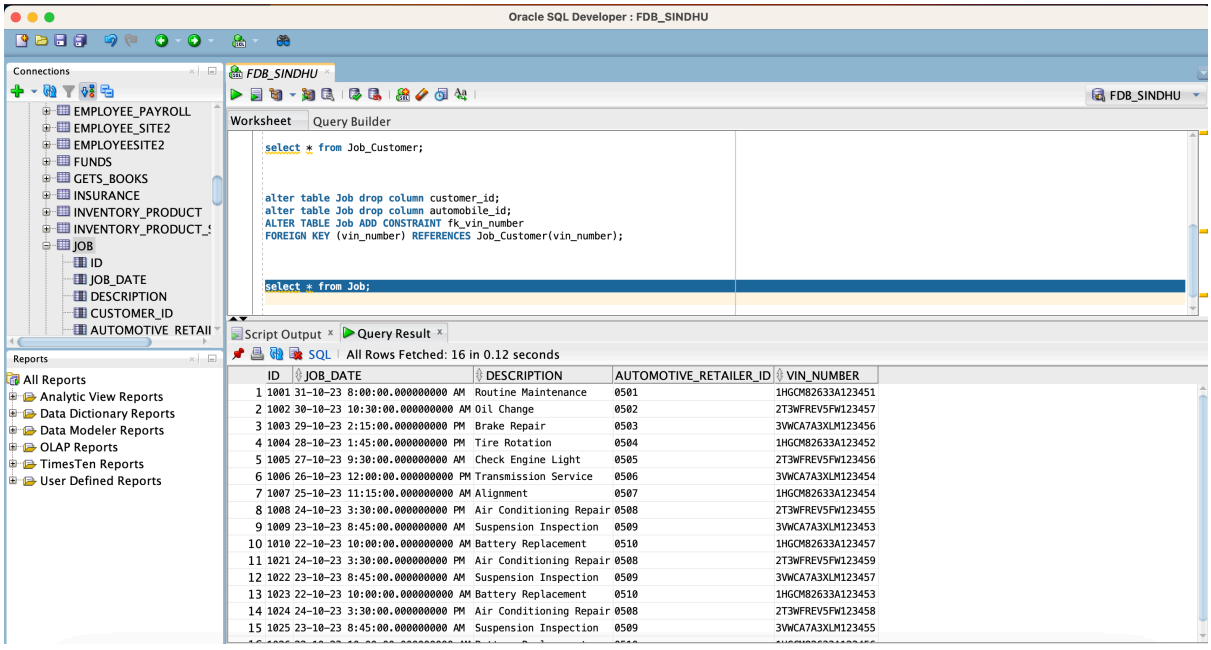
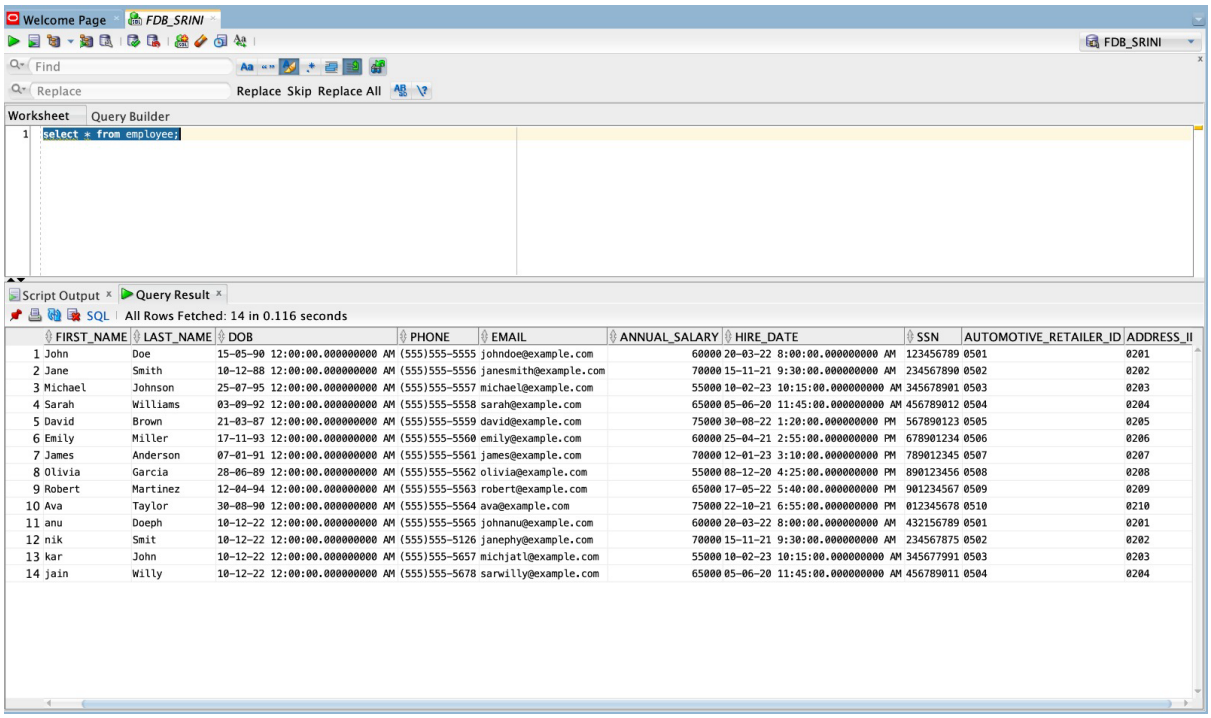


Figure-16: After altering the table.

Employee:

Decomposition implementation on employee table.

Before Decomposition:



	FIRST_NAME	LAST_NAME	DOB	PHONE	EMAIL	ANNUAL_SALARY	HIRE_DATE	SSN	AUTOMOTIVE_RETAILER_ID	ADDRESS_II
1	John	Doe	15-05-90 12:00:00.000000000 AM	(555)555-5555	johndoe@example.com	60000	20-03-22 8:00:00.000000000 AM	123456789	0501	0201
2	Jane	Smith	10-12-88 12:00:00.000000000 AM	(555)555-5556	janesmith@example.com	70000	15-11-21 9:30:00.000000000 AM	234567890	0502	0202
3	Michael	Johnson	25-07-95 12:00:00.000000000 AM	(555)555-5557	michael@example.com	55000	10-02-23 10:15:00.000000000 AM	345678901	0503	0203
4	Sarah	Williams	03-09-92 12:00:00.000000000 AM	(555)555-5558	sarah@example.com	65000	05-06-20 11:45:00.000000000 AM	456789012	0504	0204
5	David	Brown	21-03-87 12:00:00.000000000 AM	(555)555-5559	david@example.com	75000	30-08-22 1:20:00.000000000 PM	567890123	0505	0205
6	Emily	Miller	17-11-93 12:00:00.000000000 AM	(555)555-5560	emily@example.com	60000	25-04-21 2:55:00.000000000 PM	678901234	0506	0206
7	James	Anderson	07-01-91 12:00:00.000000000 AM	(555)555-5561	james@example.com	70000	12-01-23 3:10:00.000000000 PM	789012345	0507	0207
8	Olivia	Garcia	28-06-89 12:00:00.000000000 AM	(555)555-5562	olivia@example.com	55000	08-12-20 4:25:00.000000000 PM	890123456	0508	0208
9	Robert	Martinez	12-04-94 12:00:00.000000000 AM	(555)555-5563	robert@example.com	65000	17-05-22 5:40:00.000000000 PM	901234567	0509	0209
10	Ava	Taylor	30-08-90 12:00:00.000000000 AM	(555)555-5564	ava@example.com	75000	22-10-21 6:55:00.000000000 PM	012345678	0510	0210
11	anu	Doeph	10-12-22 12:00:00.000000000 AM	(555)555-5565	johnanu@example.com	60000	20-03-22 8:00:00.000000000 AM	432156789	0501	0201
12	nik	Smit	10-12-22 12:00:00.000000000 AM	(555)555-5126	janephy@example.com	70000	15-11-21 9:30:00.000000000 AM	234567875	0502	0202
13	kar	John	10-12-22 12:00:00.000000000 AM	(555)555-5657	michjatl@example.com	55000	10-02-23 10:15:00.000000000 AM	345677991	0503	0203
14	jain	Willy	10-12-22 12:00:00.000000000 AM	(555)555-5678	sarwilly@example.com	65000	05-06-20 11:45:00.000000000 AM	456789011	0504	0204

Figure-17: Before Decomposition.

Explanation: The Query **“select * from employee;”** is used to display the complete employee table data.

After Decomposition:

Creation of SSN table:

“CREATE TABLE ssn (
SSN RAW(16) NOT NULL PRIMARY KEY,
FIRST_NAME VARCHAR2(100) NOT NULL,
LAST_NAME VARCHAR2(100) NOT NULL,
DOB TIMESTAMP(6) NOT NULL,
GENDER VARCHAR2(10) NOT NULL,
PHONE VARCHAR2(20) NOT NULL
)”

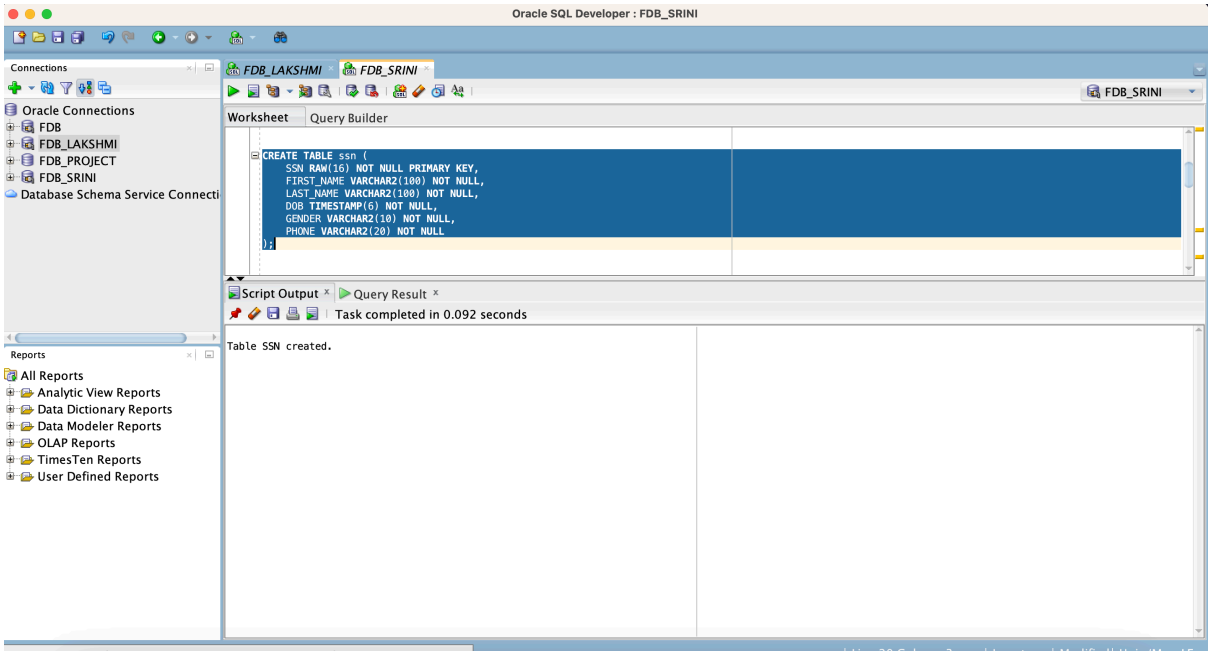


Figure-18: Creating new decomposed table.

Explanation: The ssn table is designed to store individual information. The primary key, ssn (RAW (16)), enforces uniqueness and non-null values. Additional columns include first_name, last_name (VARCHAR2(100)), dob (TIMESTAMP), and gender (VARCHAR2(10)), all with non-null constraints. The phone column (VARCHAR2(20)) has unique and non-null constraints and adheres to a specific format through a CHECK constraint using a regular expression. This structure ensures data integrity and uniqueness within the table.

SSN Data Migration:

"INSERT INTO ssn (ssn, first_name, last_name, dob, gender, phone) SELECT ssn, first_name, last_name, dob, 'MALE' gender, phone FROM Employee;"

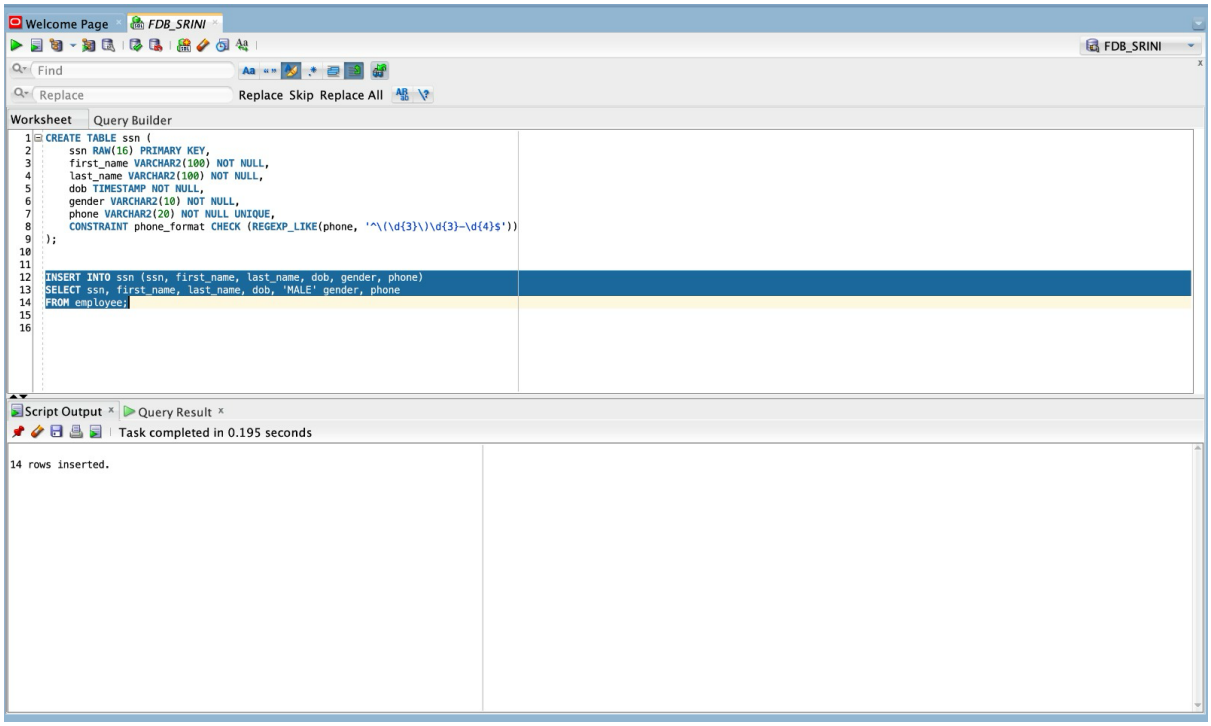


Figure-18: Data Migration.

Explanation: As this SSN table is decomposed from Employee table. We need to migrate the data from Employee table to SSN table.

SSN After Data Migration:

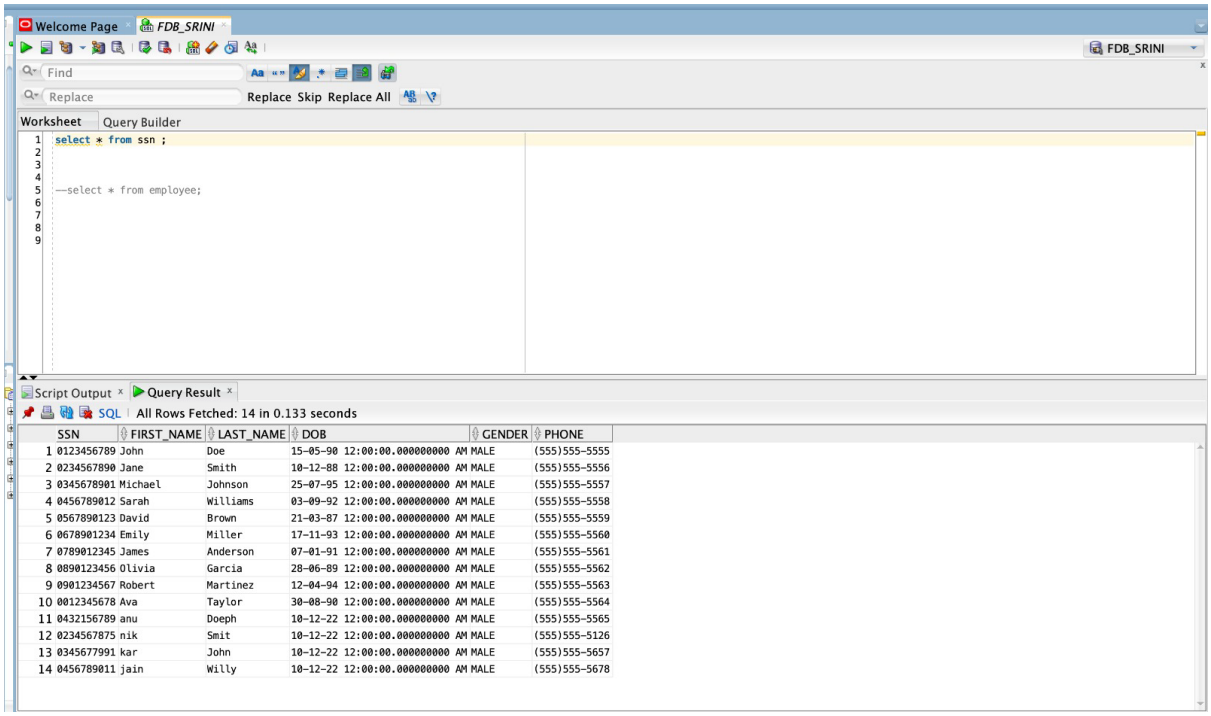


Figure-18: After data migration.

Altering the Employee Table:

Now, after migration we will drop the redundant columns in Employee table. Below Screenshot represents the dropping of columns and making 'SSN' column as the foreign key referencing the 'ssn' column in SSN table.

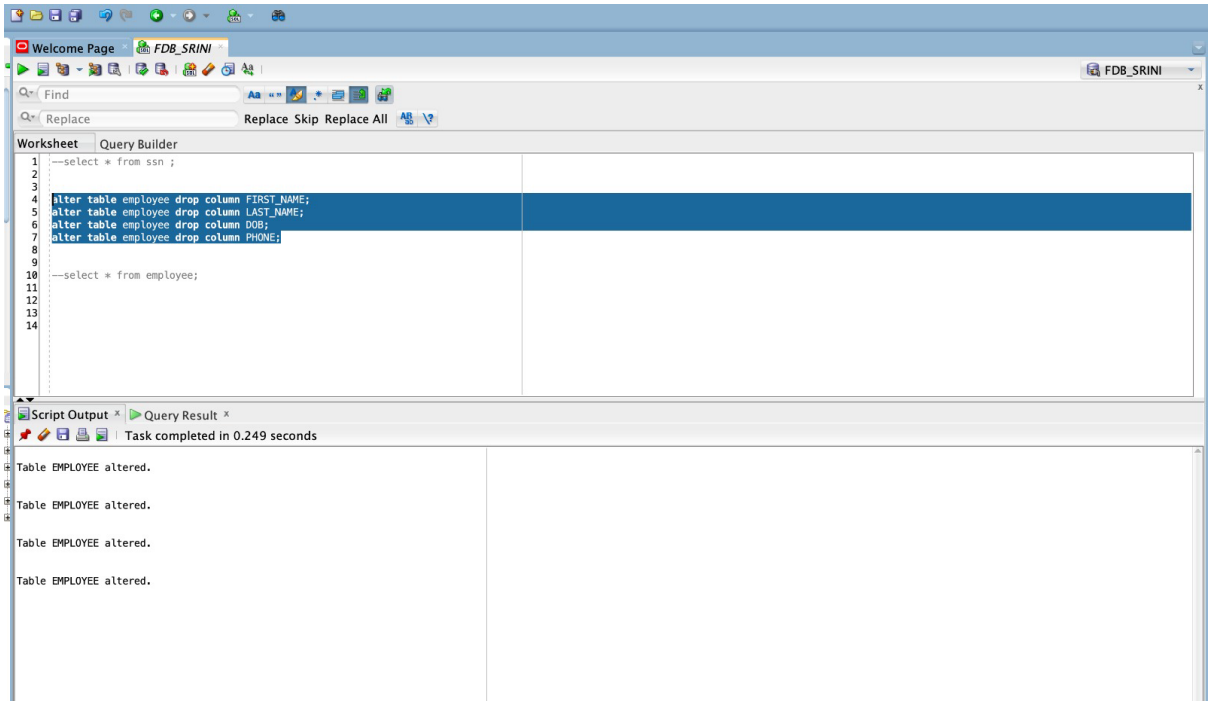


Figure-18: Altering the table.

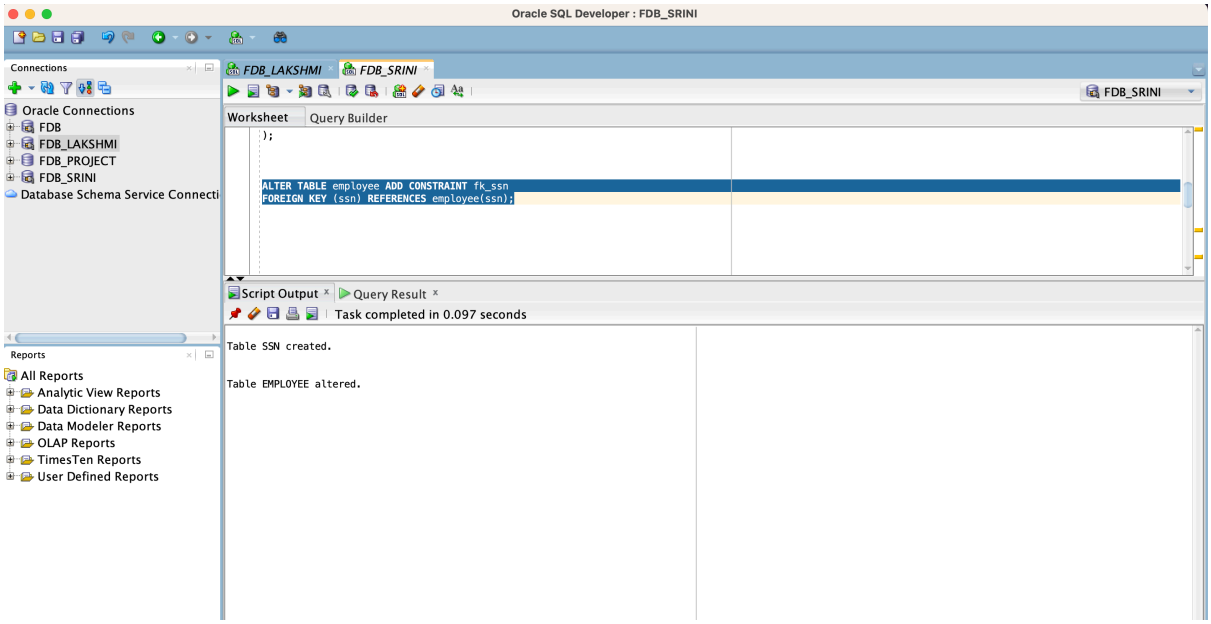
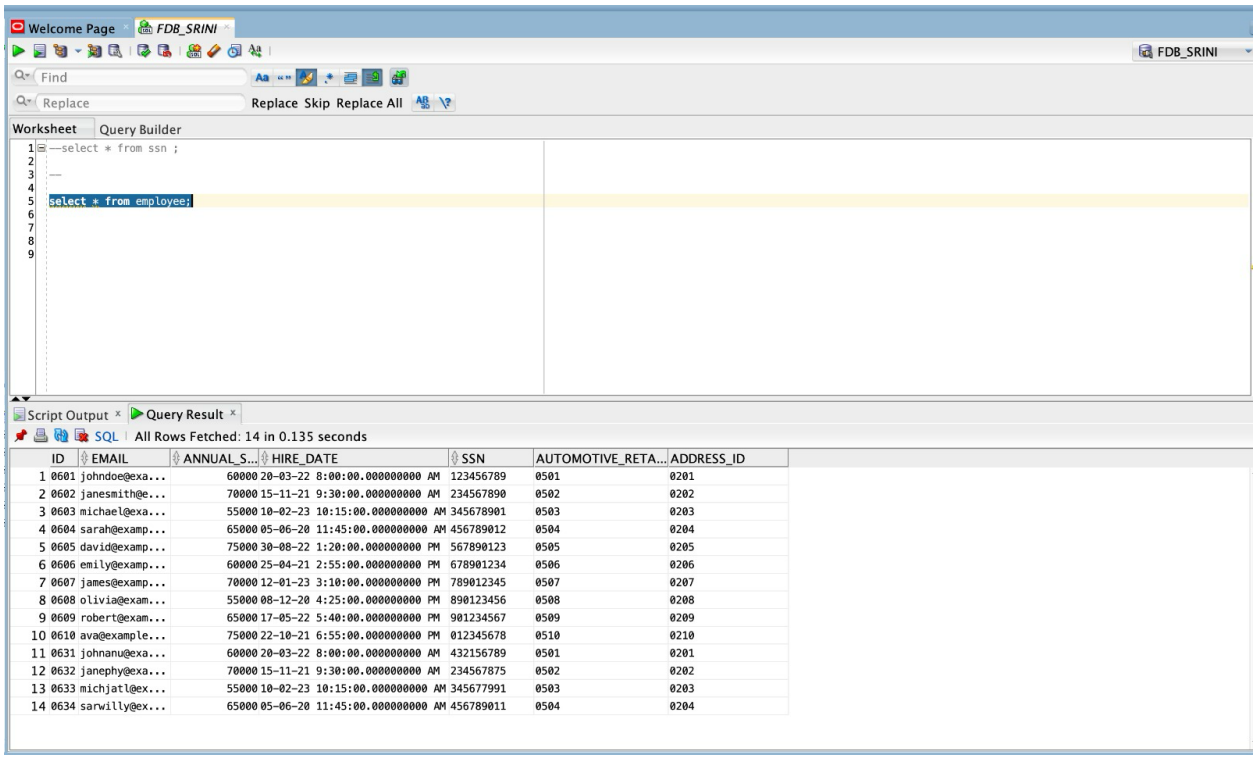


Figure-19: Altering the table.

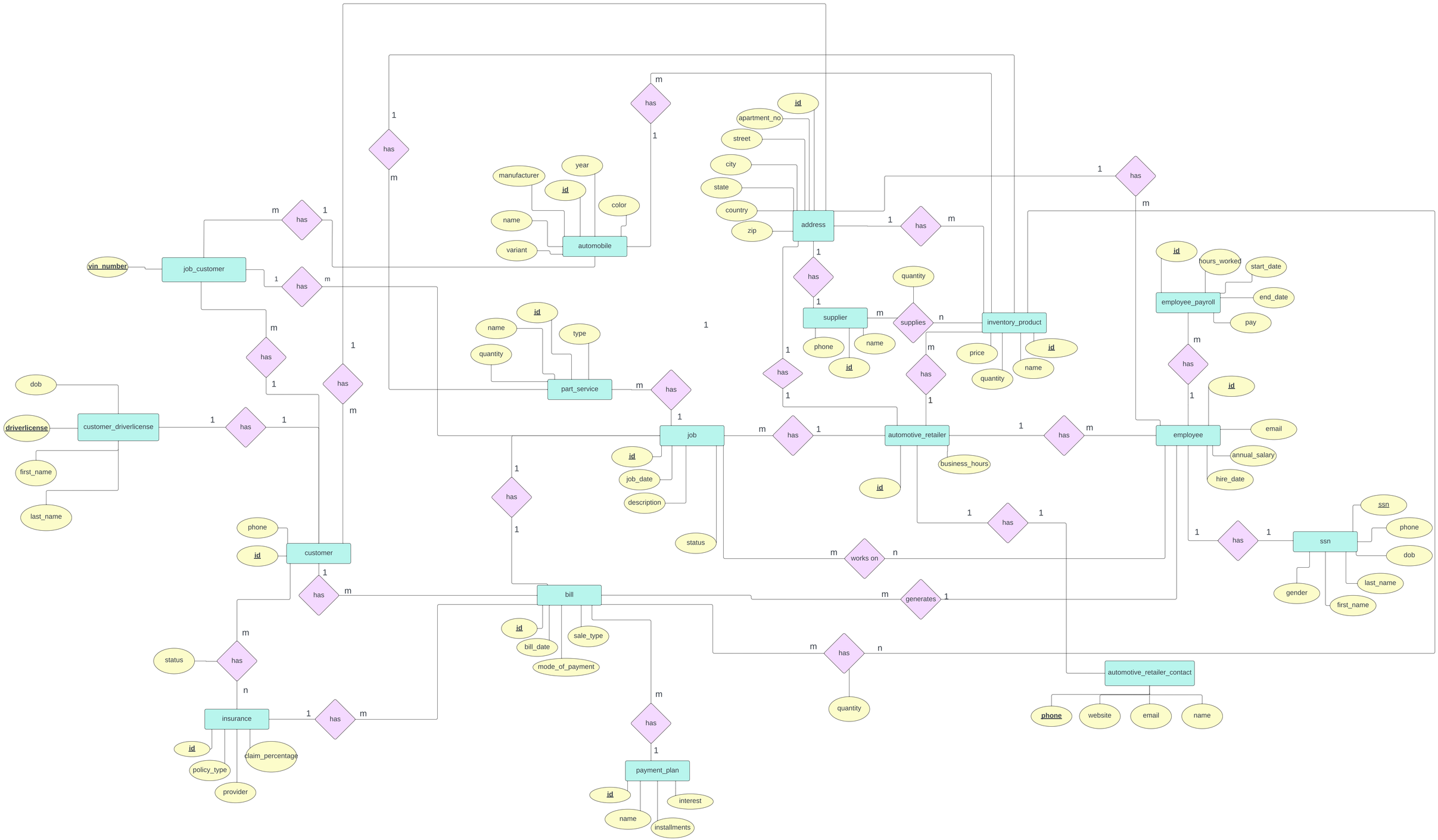
Employee table after migration:



ID	EMAIL	ANNUAL_S...	HIRE_DATE	SSN	AUTOMOTIVE_RETA...	ADDRESS_ID
1	0601	johndoe@exa...	60000	20-03-22 8:00:00.000000000 AM	123456789	0501	0201
2	0602	janesmith@e...	70000	15-11-21 9:30:00.000000000 AM	234567890	0502	0202
3	0603	michael@exa...	55000	10-02-23 10:15:00.000000000 AM	345678901	0503	0203
4	0604	sarah@examp...	65000	05-06-20 11:45:00.000000000 AM	456789012	0504	0204
5	0605	david@examp...	75000	30-08-22 1:20:00.000000000 PM	567890123	0505	0205
6	0606	emily@examp...	60000	25-04-21 2:55:00.000000000 PM	678901234	0506	0206
7	0607	james@examp...	70000	12-01-23 3:10:00.000000000 PM	789012345	0507	0207
8	0608	olivia@exam...	55000	08-12-20 4:25:00.000000000 PM	890123456	0508	0208
9	0609	robert@exam...	65000	17-05-22 5:40:00.000000000 PM	901234567	0509	0209
10	0610	ava@example...	75000	22-10-21 6:55:00.000000000 PM	012345678	0510	0210
11	0631	johnanu@exa...	60000	20-03-22 8:00:00.000000000 AM	432156789	0501	0201
12	0632	janephy@exa...	70000	15-11-21 9:30:00.000000000 AM	234567875	0502	0202
13	0633	michjatl@ex...	55000	10-02-23 10:15:00.000000000 AM	345677991	0503	0203
14	0634	sarwilly@ex...	65000	05-06-20 11:45:00.000000000 AM	456789011	0504	0204

Figure-20: Displayed the employee table after migration.

Entity Relationship Diagram



Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
automotive_retailer	id	RAW(16)	PRIMARY KEY	NA	automotive_retailer has one to many relationship with employee, automotive_retailer has one to many relationship with inventory_product, automotive_retailer has one to one relationship with address, automotive_retailer has one to many relationship with job automotive_retailer has one to one relationship with automotive_retailer_cor
	phone	VARCHAR2(20)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (phone, '^\ (\d{3}\) \d{3}-\d {4}$'))	invalid_automotive_retailer_phone	
	email	VARCHAR2(255)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (email, '^[A-Za-z0-9. _%+-]+@[A-Za-z0-9.-] +\.[A-Za-z]{2,}$'))	invalid_automotive_retailer_email	
	website	VARCHAR2(255)	NOT NULL, UNIQUE	NA	
	business_hours	VARCHAR2(20)	NOT NULL	NA	
	manager_id	RAW(16)	NOT NULL, FOREIGN KEY (manager_id) REFERENCES employee(id)	NA	
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
	name	VARCHAR2(30)	NOT NULL	NA	
inventory_product	id	RAW(16)	PRIMARY KEY	NA	inventory_product has many to many relationship with supplier, inventory_product has many to one realationship with automotive_retailer, inventory_product has many to many relationship with bill, inventory_product has one to many relationship with part_service, inventory_product has many to one relationship with automobile, inventory_product has many to one relationship with address
	name	VARCHAR2(100)	NOT NULL	NA	
	quantity	INTEGER	NOT NULL	NA	
	price	INTEGER	NOT NULL	NA	
	automotive_retailer_id	RAW(16)	NOT NULL, FOREIGN KEY (automotive_retailer_id) REFERENCES automotive_retailer(id)	NA	
	automobile_id	RAW(16)	"NOT NULL, FOREIGN KEY (automobile_id) REFERENCES automobile(id)"	NA	
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
automobile	id	RAW(16)	PRIMARY KEY	NA	automobile has one to many relationship with inventory_product, automobile has one to many relationship with job
	manufacturer	VARCHAR2(100)	NOT NULL	NA	
	name	VARCHAR2(100)	NOT NULL	NA	
	variant	VARCHAR2(100)	NOT NULL	NA	
	year	VARCHAR2(10)	NOT NULL	NA	
	color	VARCHAR2(100)	NOT NULL	NA	
	id	RAW(16)	PRIMARY KEY	NA	
	name	VARCHAR2(100)	NOT NULL	NA	

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
supplier	phone	VARCHAR2(20)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (phone, '^\ (\d{3}\) \d{3}-\d {4}$'))	invalid_supplier_phone	supplier has many to many relationship with inventory_product, supplier has one to one relationship with address
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
Employee	id	RAW(16)	PRIMARY KEY	NA	employee has one to many relationship with employee_payroll, employee has many to one relationship with automotive_retailer, employee has one to many relationship with bill, employee has many to one relationship with address, employee has many to many relationship with job, employee has one to one relation with ssn
	email	VARCHAR2(255)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (email, '^[A-Za-z0-9. _%+-]+@[A-Za-z0-9.-] +\.[A-Za-z]{2,}$'))	invalid_employee_email	
	annual_salary	INTEGER	NOT NULL	NA	
	ssn	RAW(16)	NOT NULL, FOREIGN KEY (ssn) REFERENCES (ssn)	NA	
	automotive_retailer_id	RAW(16)	NOT NULL, FOREIGN KEY (automotive_retailer_id) REFERENCES automotive_retailer(id)	NA	
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
	hire_date	TIMESTAMP	NOT NULL	NA	
employee_payroll	id	RAW(16)	PRIMARY KEY	NA	employee_payroll has many to one relationship with employee
	hours_worked	NUMBER	NOT NULL	NA	
	start_date	TIMESTAMP	NOT NULL	NA	
	end_date	TIMESTAMP	NOT NULL	NA	
	pay	NUMBER	NOT NULL	NA	
	employee_id	RAW(16)	NOT NULL, FOREIGN KEY (employee_id) REFERENCES employee(id)	NA	
address	id	RAW(16)	PRIMARY KEY	NA	address has one to many relationship with employee, address has one to many relationship with inventory_product, address has one to one relationship with automotive_retailer, address has one to one relationship with supplier, address has one to many relationship with customer
	apartment_no	NUMBER	NOT NULL	NA	
	street	VARCHAR2(100)	NOT NULL	NA	
	city	VARCHAR2(100)	NOT NULL	NA	
	state	VARCHAR2(50)	NOT NULL	NA	
	country	VARCHAR2(50)	NOT NULL	NA	
	zip	CHAR(5)	NOT NULL	NA	
	id	RAW(16)	PRIMARY KEY	NA	
	date	TIMESTAMP	NOT NULL	NA	

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
bill	mode_of_payment	CHAR(4)	NOT NULL, CHECK (mode_of_payment IN ('CARD', 'CASH'))	invalid_mode_of_payment	bill has many to one relationship with employee, bill has many to many relationship with inventory_product, bill has many to one relationship with insurance, bill has many to one relationship with customer, bill has one to one relationship with job, bill has many to one relationship with payment_plan
	insurance_id	RAW(16)	NOT NULL, FOREIGN KEY (insurance_id) REFERENCES insurance(id)	NA	
	customer_id	RAW(16)	NOT NULL, FOREIGN KEY (customer_id) REFERENCES customer(id)	NA	
	job_id	RAW(16)	NOT NULL, FOREIGN KEY (job_id) REFERENCES job(id)	NA	
	employee_id	RAW(16)	NOT NULL, FOREIGN KEY (employee_id) REFERENCES employee(id)	NA	
	sale_type	CHAR(7)	NOT NULL, CHECK (mode_of_payment IN ('ONLINE', 'OFFLINE'))	invalid_sale_type	
	payment_plan_id	RAW(16)	NOT NULL, FOREIGN KEY (payment_plan_id) REFERENCES payment_plan(id)	NA	
job	id	RAW(16)	PRIMARY KEY	NA	job has one to one realtionship with bill, job has many to one realtionship with automotive_retailer, job has many to many relationship with employee, job has one to many realtionship with part_service,
	date	TIMESTAMP	NOT NULL	NA	
	description	VARCHAR2(255)	NOT NULL	NA	
	customer_id	RAW(16)	NOT NULL, FOREIGN KEY (customer_id) REFERENCES customer(id)	NA	
	automotive_retailer_id	RAW(16)	NOT NULL, FOREIGN KEY (automotive_retailer_id) REFERENCES automotive_retailer(id)	NA	
	vin_number	VARCHAR2(100)	NOT NULL, CHECK()	NA	
	automobile_id	RAW(16)	NOT NULL, FOREIGN KEY (automobile_id) REFERENCES automobile(id)	NA	
	job_status	VARCHAR2(20)		NA	
	id	RAW(16)	PRIMARY KEY	NA	
	name	VARCHAR2(100)	NOT NULL	NA	
	quantity	INTEGER	NOT NULL	NA	
	job_id	RAW(16)	NOT NULL, FOREIGN KEY (job_id) REFERENCES job(id)	NA	part_service has many to one relationship with job

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
part_service	inventory_product_id	RAW(16)	NOT NULL, FOREIGN KEY (inventory_product_id) REFERENCES inventory_product(id)	NA	part_service has many to one relationship with job, part_service has one to one relationship with inventary_product
	type	CHAR(7)	NOT NULL, CHECK (mode_of_payment IN ('SERVICE', 'PART'))	invalid_part_service_type	
payment_plan	id	RAW(16)	PRIMARY KEY	NA	payment_plan has one to many relationship with bill
	name	VARCHAR2(100)	NOT NULL	NA	
	installments	INTEGER	NOT NULL	NA	
	interest	INTEGER	NOT NULL	NA	
customer	id	RAW(16)	PRIMARY KEY	NA	customer has many to many relationship with insurance, customer has one to many relationship with job, customer has many to one relationship with address, customer has one to many relationship with bills, customer has one to one relationship with customer_driverlicense customer has one to many relationship with job_customer
	first_name	VARCHAR2(100)	NOT NULL	NA	
	last_name	VARCHAR2(100)	NOT NULL	NA	
	dob	TIMESTAMP	NOT NULL	NA	
	driverlicense	VARCHAR2(50)	UNIQUE, NOT NULL	NA	
	phone	VARCHAR2(20)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (phone, '^\ (\d{3}\) \d{3}-\d {4}$'))	invalid_customer_phone	
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
insurance	id	RAW(16)	PRIMARY KEY	NA	insurance has many to many relationship with customer, insurance has one to many relationship with bill
	policy_type	VARCHAR2(100)	NOT NULL	NA	
	provider	VARCHAR2(100)	NOT NULL	NA	
	claim_percentage	INTEGER	NOT NULL	NA	
inventory_product_supplier	inventory_product_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (inventory_product_id) REFERENCES inventory_product(id)"	NA	NA
	supplier_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (supplier_id) REFERENCES supplier(id)	NA	
	quantity	INTEGER	NOT NULL	NA	
	bill_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (bill_id) REFERENCES bill(id)"	NA	

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
bill_inventory_product	inventory_product_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (inventory_product_id) REFERENCES inventory_product(id)"	NA	NA
	quantity	INTEGER	NOT NULL	NA	
job_employee	job_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (job_id) REFERENCES job(id)"	NA	NA
	employee_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (employee_id) REFERENCES employee(id)"	NA	
customer_insurance	customer_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (customer_id) REFERENCES customer(id)"	NA	NA
	insurance_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (insurance_id) REFERENCES insurance(id)"	NA	
	status	CHAR(8)	NOT NULL	NA	
ssn	ssn	RAW(16)	PRIMARY KEY, NOT NULL	NA	ssn has one to one relationship with employee
	first_name	VARCHAR2(100)	NOT NULL	NA	
	last_name	VARCHAR2(100)	NOT NULL	NA	
	dob	TIMESTAMP	NOT NULL	NA	
	gender	VARCHAR2(10)	NOT NULL	NA	
	phone	VARCHAR2(20)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (phone, '^\ (\d{3}\) \d{3}-\d {4}$'))	invalid_employee_phone	
customer_driverlicense	driverlicense	VARCHAR2(50)	UNIQUE, NOT NULL, PRIMARY KEY	NA	customer_drivelicense has one to one relationship with customer
	first_name	VARCHAR2(100)	NOT NULL	NA	
	last_name	VARCHAR2(100)	NOT NULL	NA	
	dob	TIMESTAMP	NOT NULL	NA	
Job_Customer	vin_number	vin_number	VARCHAR2(100)	NOT NULL, CHECK()	Job_Customer has many to one relationship with Automobile Job_Customer has many to one relationship with Customer Job_Customer has one to many relationship with Job
	customer_id	RAW(16)	NOT NULL, FOREIGN KEY (customer_id) REFERENCES customer(id)	NA	

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
	automobile_id	RAW(16)	NOT NULL, FOREIGN KEY (automobile_id) REFERENCES automobile(id)	NA	Job_Customer has one to many relationship with Job