FUTURE DESIGN

Car Dealership Management System

Group 6

Vijay Venkatesan
Anamika Das
Ninad Patil
Yao-Hui Tseng
Krish Sonani
Ishan Waghmare
Nihal Sharma

TABLE OF CONTENTS

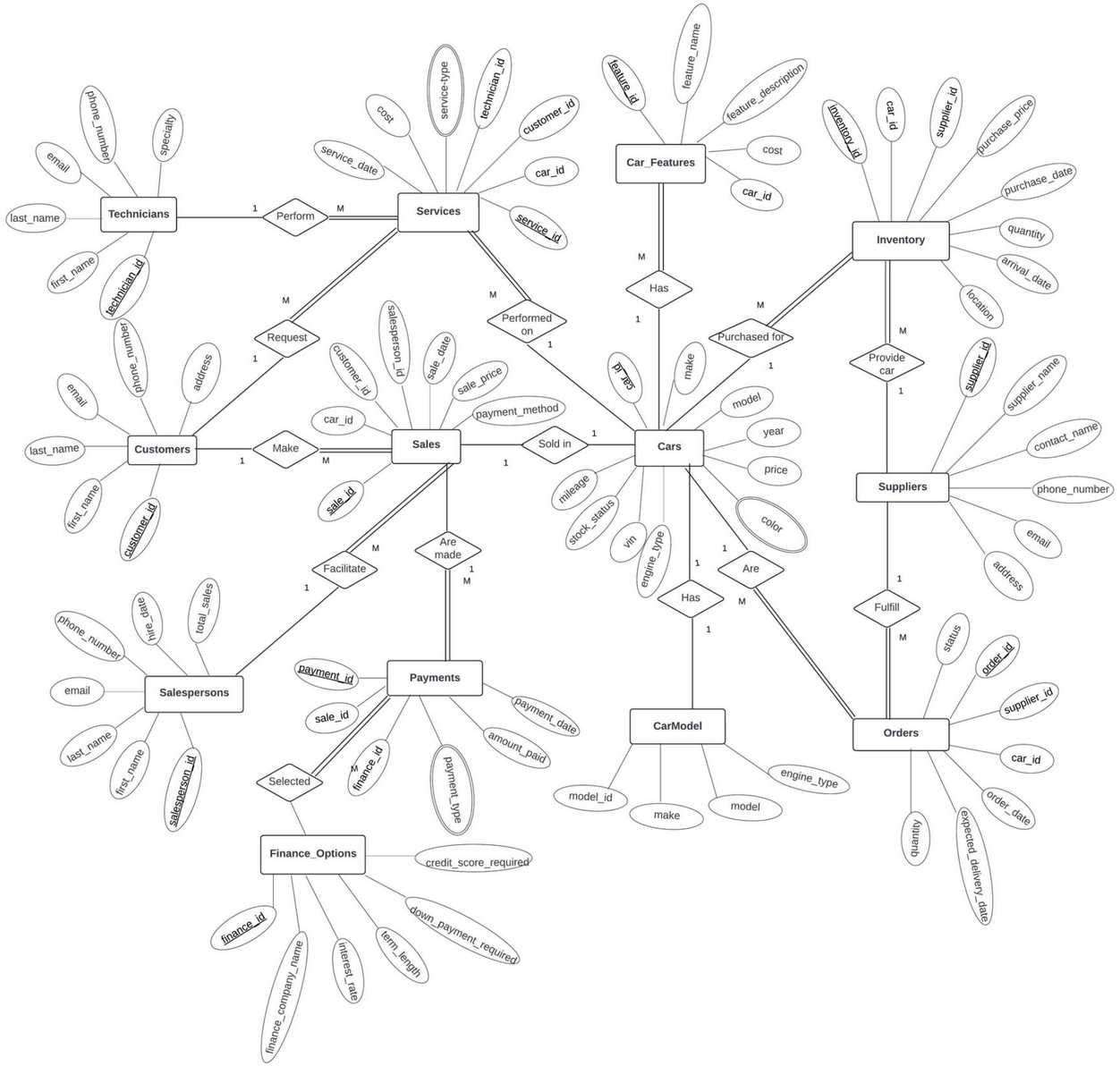
INTRODUCTION

- Database for a car dealership
- Streamlines processes like sales, service, customer management, and inventory.
- Centralizes financial options, service records, customer data, and vehicle info.
- Enables real-time access for reporting and strategy improvements.

GOALS

- Real-time visibility of car availability and status.
- Track sales, services, and client preferences.
- Manage financing, promotions, and payment history.
- Schedule and track maintenance and repairs.
- Provide insights on suppliers, technicians, and sales trends.

ENTITY RELATIONSHIP DIAGRAM



- Entity
- Attribute
- Multivalued Attribute
- Primary Key
- Relationship
- Cardinality

INDIVIDUAL COMPONENTS

ITS SQL TIME!!!

Salesperson 1: Vijay Venkatesan

Query 1: Total Number of Cars in Inventory That Are Still 'In Stock'

Objective: Helps track how many cars are ready for sale, aiding inventory management and sales forecasting

```
533     #Count the Total Number of Cars in Inventory That Are Still 'In Stock
534 •   SELECT COUNT(*) AS TotalInStockCars
535     FROM Cars
536     WHERE stock_status = 'In Stock';
537
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TotalInStockCars
62

Salesperson 1: Vijay Venkatesan

Query 2: Finding the highest price for an electric vehicle

Objective: Provides insights into high-value vehicles in specific categories, supporting targeted marketing and pricing strategies.

```
538     #Find the Maximum Price of Cars in a Specific Engine Type
539 •   SELECT MAX(price) AS MaxPrice
540     FROM Cars c
541     JOIN CarModels cm ON c.model_id = cm.model_id
542     WHERE cm.engine_type = 'Electric';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MaxPrice
80000.00

Salesperson 1: Vijay Venkatesan

Query 3: Finding total revenue for salespersons that sold more cars

Objective: Identify top-performing salespersons, enabling the dealership to reward success and strategize sales training.

```
549     #Find the Total Revenue Generated by Salespersons Who Sold More Than 5 Cars.
550 •   SELECT s.salesperson_id, SUM(s.sale_price) AS TotalRevenue
551     FROM Sales s
552     GROUP BY s.salesperson_id
553     HAVING COUNT(s.sale_id) > 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

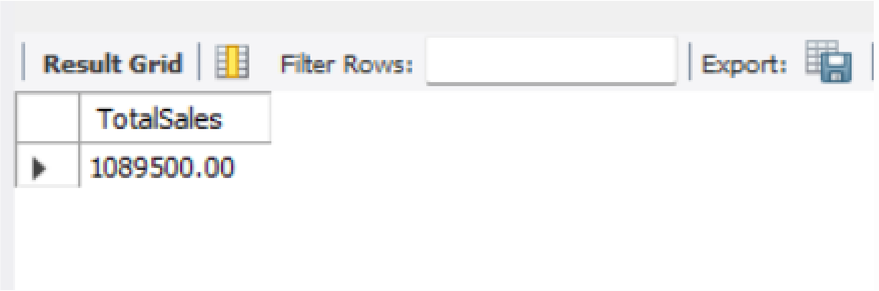
salesperson_id	TotalRevenue
1	139000.00

Interactor: Anamika Das

Query 1: Calculate the total sales by summing up the sales for all cars using a stored procedure.
Objective: To encapsulate repetitive tasks and reuse it which improves performance and productivity.

```
DELIMITER //

CREATE PROCEDURE CalculateTotalSales()
BEGIN
    DECLARE total_sales DECIMAL(15, 2);

    -- Calculate total sales by summing the sale_price from Sales table
    SELECT SUM(sale_price) INTO total_sales
    FROM Sales;

    -- Return the total sales
    SELECT total_sales AS TotalSales;
END //

DELIMITER ;

CALL CalculateTotalSales();
```

Output:



Interactor: Anamika Das

Query 2: The total amount paid for each sale, along with the finance company and interest rate, where the payment exceeds 5000.
Objective: To provide a concise view of sales, their associated payments, and finance company details while applying basic filtering and aggregation.

```sql
SELECT
    S.sale_id,
    SUM(P.amount_paid) AS TotalAmountPaid,
    F.finance_company_name,
    F.interest_rate
FROM
    Sales S JOIN
    Payments P ON S.sale_id = P.sale_id
JOIN
    FinanceOptions F ON S.sale_id = F.finance_id
GROUP BY
    S.sale_id, F.finance_company_name, F.interest_rate
HAVING
    SUM(P.amount_paid) > 5000  -- Only include sales where total amount paid is greater than 5000
ORDER BY
    TotalAmountPaid DESC;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sale_id	TotalAmountPaid	finance_company_name	interest_rate
7	95000.00	EasyAuto Finance	4.20
1	50000.00	Bank of Auto	2.50
6	47000.00	PrimeAuto Loans	3.50
2	42000.00	Finance4U	3.00
3	38000.00	AutoCredit Union	3.20
5	38000.00	National Car Finance	2.80
4	22000.00	VehicleLoans Corp.	4.00

Interactor: Anamika Das

Query 3: Create a view that combines detailed information about payment types, amounts paid, and financing details like pending or paid for cars purchased.

Objective: Views provide a simplified and Custom data representation, enhance security by restricting access to specific columns or rows of a table.

```sql
CREATE VIEW DetailedPaymentFinanceView AS
SELECT
    p.payment_id AS PaymentID,
    p.payment_type AS PaymentType,
    p.amount_paid AS AmountPaid,
    f.finance_id AS FinanceID,
    f.interest_rate AS InterestRate,
    (1 + (f.interest_rate / 100)) AS TotalPayableAmount,
    CASE
        WHEN SUM(p.amount_paid) OVER (PARTITION BY f.finance_id) >=
             (1 + (f.interest_rate / 100))
        THEN 'Paid in Full'
        ELSE 'Pending'
    END AS PaymentStatus,
    SUM(p.amount_paid) OVER (PARTITION BY f.finance_id) AS TotalPaymentsMade
FROM payments p LEFT JOIN financeoptions f
ON p.payment_id = f.finance_id;

SELECT * FROM DetailedPaymentFinanceView;
```

Output:

PaymentID	PaymentType	AmountPaid	FinanceID	InterestRate	TotalPayableAmount	PaymentStatus	TotalPaymentsMade
20	Cash	20000.00	NULL	NULL	NULL	Pending	471000.00
21	Debit Card	5000.00	NULL	NULL	NULL	Pending	471000.00
22	Bank Transfer	25000.00	NULL	NULL	NULL	Pending	471000.00
23	Credit Card	30000.00	NULL	NULL	NULL	Pending	471000.00
24	Cash	9000.00	NULL	NULL	NULL	Pending	471000.00
25	Debit Card	16000.00	NULL	NULL	NULL	Pending	471000.00
1	Cash	35000.00	1	2.50	1.025000	Paid in Full	35000.00
2	Bank Transfer	20000.00	2	3.00	1.030000	Paid in Full	20000.00
3	Credit Card	20000.00	3	3.20	1.032000	Paid in Full	20000.00
4	Credit Card	10000.00	4	4.00	1.040000	Paid in Full	10000.00
5	Cash	18000.00	5	2.80	1.028000	Paid in Full	18000.00
6	Credit Card	42000.00	6	3.50	1.035000	Paid in Full	42000.00
7	Bank Transfer	70000.00	7	4.20	1.042000	Paid in Full	70000.00

Salesperson 3: Ninad Patil

Query 1: 10 Cars with the highest profit margins

Objective: To get an idea which are the top 10 cars that generate the most amount of profit

```
SELECT
    cm.make,
    cm.model,
    c.year,
    s.sale_price,
    i.purchase_price,
    (s.sale_price - i.purchase_price) AS profit_margin

FROM Cars c
JOIN CarModels cm ON c.model_id = cm.model_id
JOIN Sales s ON c.car_id = s.car_id
JOIN Inventory i ON c.car_id = i.car_id
ORDER BY profit_margin DESC
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

make	model	year	sale_price	purchase_price	profit_margin
Tesla	Model S	2022	80000.00	26000.00	54000.00
Tesla	Model 3	2022	73000.00	20500.00	52500.00
Tesla	Model 3	2022	70000.00	20500.00	49500.00
BMW	X5	2021	62000.00	24500.00	37500.00
BMW	X5	2021	59000.00	24500.00	34500.00
Mercedes	C-Class	2019	54000.00	22500.00	31500.00
Mercedes	C-Class	2019	54000.00	22500.00	31500.00
Chevrolet	Tahoe	2021	51000.00	21500.00	29500.00
Chevrolet	Camaro	2021	40000.00	18500.00	21500.00
Audi	A4	2020	52000.00	43000.00	9000.00

Query 2: Analyze car stock status: List all models with cars that remain unsold for more than 2 years after manufacturing.

Objective: To check which car models in the stock are older than 2 years and send them back to the manufacturer

```
SELECT
    c.vin,
    cm.make,
    cm.model,
    c.year AS manufacture_year,
    c.stock_status,
    DATEDIFF(CURDATE(),DATE(CONCAT(c.year, '-01-01'))) /
365 AS years_since_manufacture

FROM Cars c
JOIN CarModels cm ON c.model_id = cm.model_id
WHERE c.stock_status = 'In Stock' AND
DATEDIFF(CURDATE(), DATE(CONCAT(c.year, '-01-01'))) >
730;
```

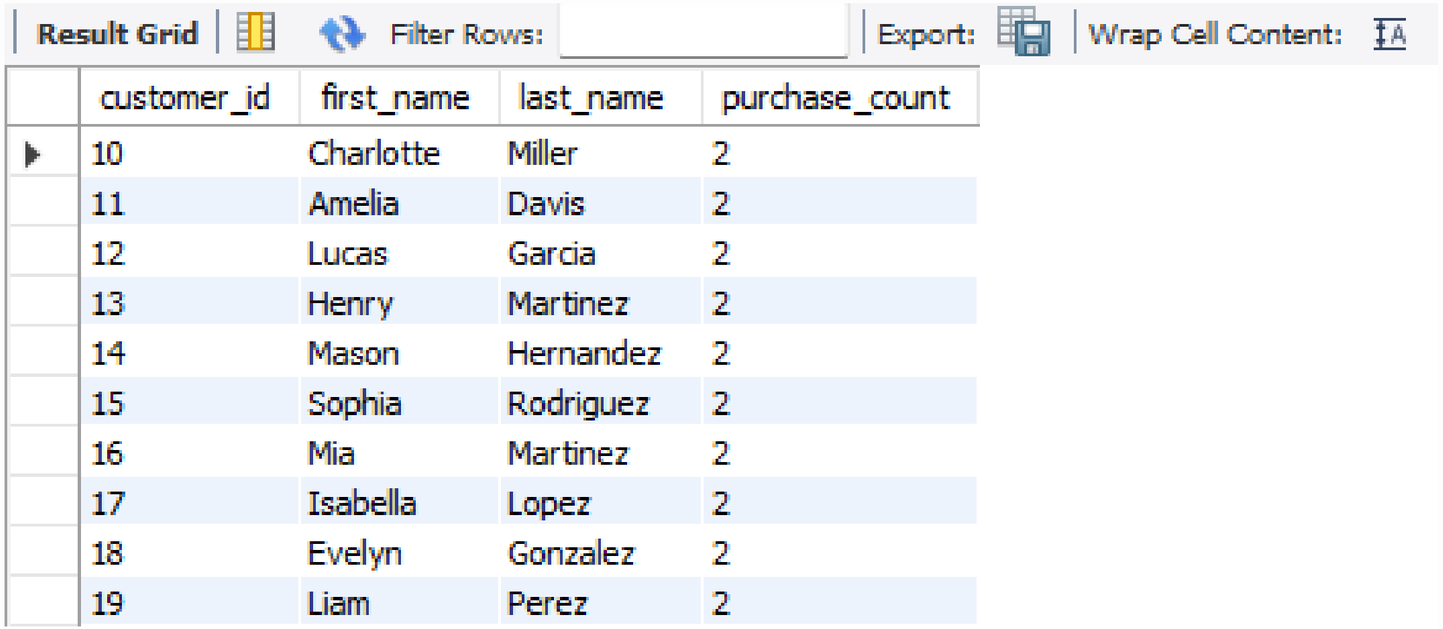
sult Grid | Filter Rows: | Export: | Wrap Cell Content:

vin	make	model	manufacture_year	stock_status	years_since_manufacture
JTMZK33V876009879	Toyota	Corolla	2020	In Stock	4.9315
5YJSA1E25FF100001	Tesla	Model S	2022	In Stock	2.9288
19XFC2F59JE216546	Honda	Civic	2019	In Stock	5.9315
WBAXH5C59F0K12345	BMW	X5	2021	In Stock	3.9288
WAUZZZF48HA123456	Audi	A4	2020	In Stock	4.9315
5YJ3E1EA1JF123456	Tesla	Model 3	2022	In Stock	2.9288
1FMCU9GX5HUA12345	Ford	Escape	2020	In Stock	4.9315
1GNFK13027J123456	Chevrolet	Tahoe	2021	In Stock	3.9288
1N4AL3AP8DC123456	Nissan	Altima	2017	In Stock	7.9315
1MBEJXXJXPN123456	Mercedes-Benz	S-Class	2022	In Stock	2.9288
2BMWJXXJXPN123457	BMW	7 Series	2021	In Stock	3.9288
3AUDIXXJXPN123458	Audi	A8	2022	In Stock	2.9288
4PORSCHEXXJXPN123...	Porsche	Panam...	2021	In Stock	3.9288
5LEXUSXXJXPN123460	Lexus	LS	2020	In Stock	4.9315
7TESLAXXJXPN123462	Land Rover	Range ...	2022	In Stock	2.9288
8LANDROVERXXJXPN...	Maserati	Quattr...	2021	In Stock	3.9288
9MASERATIXXJXPN12...	Bentley	Contin...	2022	In Stock	2.9288
1FIATXXJXPN123491	Mini	Cooper	2022	In Stock	2.9288
2MINIXXJXPN123492	Honda	HR-V	2021	In Stock	3.9288
3HONDAHRVXXJXPN1...	Nissan	Kicks	2022	In Stock	2.9288
4NISSANKICKSXXJXP...	Toyota	C-HR	2021	In Stock	3.9288
5TOYOTACHRXXJXPN...	Hyundai	Venue	2022	In Stock	2.9288
6HYUNDAIVENUEXXJX...	Kia	Soul	2022	In Stock	2.9288
7KIASOULXXJXPN123...	Chevrolet	Spark	2021	In Stock	3.9288
8CHEVYSXXJXPN123498	Mazda	CX-30	2022	In Stock	2.9288
9MAZDACX30XXJXPN...	Subaru	Crosstrek	2021	In Stock	3.9288
0SUBARUCROSSTREK...	Volkswagen	Tiguan	2022	In Stock	2.9288
1VOLKSTIGUANXXJXP...	Chrysler	Pacifica	2021	In Stock	3.9288
2CHRYSLERPACIFICA...	Dodge	Journey	2022	In Stock	2.9288

Query 3: Find customers who have made multiple purchases

Objective: To find the customers who have bought more than 1 car to come up with some additional offers for them in the future

```
SELECT c.customer_id, c.first_name, c.last_name, COUNT(*) as purchase_count
FROM Customers c
JOIN Sales s ON c.customer_id = s.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING COUNT(*) > 1;
```



customer_id	first_name	last_name	purchase_count
10	Charlotte	Miller	2
11	Amelia	Davis	2
12	Lucas	Garcia	2
13	Henry	Martinez	2
14	Mason	Hernandez	2
15	Sophia	Rodriguez	2
16	Mia	Martinez	2
17	Isabella	Lopez	2
18	Evelyn	Gonzalez	2
19	Liam	Perez	2

Salesperson 4: Yao Hui Tseng

Query 1: List All Cars in Inventory that have yet to accumulate 8000 miles, Ordered by Their Mileage in Ascending Order.

Objective: Identify low-mileage cars that may be more attractive to buyers, optimizing inventory promotion.

```
580    #List All Cars in Inventory that have mileage less than 8000, Ordered by Their Mileage in Ascending Order
581 •  SELECT car_id, vin, mileage
582    FROM Cars
583    WHERE mileage < 8000
584    ORDER BY mileage ASC;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

car_id	vin	mileage
64	9SUBARUIMPREZAXXJXPN123529	1300
53	3JEEPRENEGADEXXJXPN123518	1300
54	4HYUNDAITUCSONXXJXPN123519	1500
35	0SUBARUCROSSTREKXXJXPN123500	1500
9	5YJ3E1EA1JF123456	1500
40	5NISSANROGUEXXJXPN123505	1500
57	2HONDACIVICXXJXPN123522	1500
26	1FIATXXJXPN123491	1500
42	2HONDAFITXXJXPN123507	1500

Salesperson 4: Yao Hui Tseng

Query 2: Find the Total Cost of Features for a Specific Car

Objective: Provide cost breakdowns for customization, helping in pricing bundled features competitively.

```
586     #Find the Total Cost of Features for a Specific Car
587 •   SELECT car_id, SUM(cost) AS TotalFeatureCost
588     FROM CarFeatures
589     WHERE car_id = 1
590     GROUP BY car_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

car_id	TotalFeatureCost
1	3500.00

Salesperson 4: Yao Hui Tseng

Query 3: Find the Minimum Sale Price for Cars Sold Using 'Cash' .

Objective: Evaluate cash sale patterns, supporting strategies to encourage cash or other preferred payment methods

```
592     #Find the Minimum Sale Price for Cars Sold Using 'Cash' Payment Method
593 •   SELECT MIN(s.sale_price) AS MinCashSalePrice
594     FROM Sales s
595     JOIN Payments p ON s.sale_id = p.sale_id
596     WHERE p.payment_type = 'Cash';
597
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MinCashSalePrice
18000.00

Salesperson 5: Nihal Sharma

-- query 1 -- Finding the top 3 customers based on total spendings at the dealership and listing details of each car they purchased.
[subquery total spendings of each customer, ordering by total spendings per customer/cars]
-- objective: salesperson is able to create customer relations with the best customers in the future.

```
SELECT
  CustomerSpending.Customer_Name,
  c.make AS Manufacturer,
  c.model AS Model,
  c.year AS Year,
  s.sale_price AS Sale_Price,
  s.sale_date AS Sale_Date
FROM
  (SELECT
    cu.customer_id,
    cu.first_name AS Customer_Name,
    SUM(s.sale_price) AS Total_Spending
   FROM
    CarDealershipDataBase.Customers cu
   JOIN
    CarDealershipDataBase.Sales s ON cu.customer_id = s.customer_id
   GROUP BY
    cu.customer_id
   ORDER BY
    Total_Spending DESC
   LIMIT 3) AS CustomerSpending
JOIN
  CarDealershipDataBase.Sales s ON CustomerSpending.customer_id = s.customer_id
JOIN
  CarDealershipDataBase.Cars c ON s.car_id = c.car_id
ORDER BY
  CustomerSpending.Total_Spending DESC, s.sale_price DESC;
```

Output:

Customer_Name	Manufacturer	Model	Year	Sale_Price	Sale_Date
Amelia	BMW	X5	2021	62000.00	2022-12-13
Amelia	Chevrolet	Tahoe	2021	22500.00	2024-03-16
Evelyn	BMW	X5	2021	59000.00	2023-07-21
Evelyn	Maserati	Quattroporte	2022	23000.00	2024-03-23
Jane	Tesla	Model S	2022	80000.00	2022-01-15

-- query 2 -- Calculating the percentage of total revenue contributed by each manufacturer to get a distributed percentage of sale.
[rounding sum of sales price as total revenue]
-- objective: dealership achieves an understanding of their sales percentages/ brand.

```
SELECT
    c.make AS Manufacturer,
    ROUND(SUM(s.sale_price), 2) AS Total_Revenue,
    ROUND((SUM(s.sale_price) / (SELECT SUM(sale_price) FROM CarDealershipDataBase.Sales)) * 100, 2) AS Revenue_Percentage
FROM
    CarDealershipDataBase.Cars c
JOIN
    CarDealershipDataBase.Sales s ON c.car_id = s.car_id
GROUP BY
    c.make
ORDER BY
    Total_Revenue DESC;
```

Output:

Manufacturer	Total_Revenue	Revenue_Percentage
Tesla	271000.00	24.87
BMW	164500.00	15.10
Chevrolet	133500.00	12.25
Mercedes	131500.00	12.07
Audi	94000.00	8.63
Kia	74000.00	6.79
Ford	61000.00	5.60
Toyota	54000.00	4.96
Honda	47000.00	4.31
Maserati	23000.00	2.11
Nissan	18000.00	1.65
Jaguar	18000.00	1.65

-- procedure (query 3) -- filtering car make, min/max sales price, year of sale to retrieve car sales data.
[4 input parameters, added dynamic filtering ability, defaulting ordering by sale date]
-- objective: providing an ability to the dealership/sales team to filter and retrieve car sales data based on inputs such as: the cars make, minimum sales price, maximum sales price, and the year of sale.

```
CREATE PROCEDURE GetFilteredSalesData(
    IN ManufacturerName VARCHAR(50),      -- Filter by car manufacturer (NULL for all manufacturers)
    IN MinSalePrice DECIMAL(10,2),        -- Minimum sale price filter (NULL for no minimum)
    IN MaxSalePrice DECIMAL(10,2),        -- Maximum sale price filter (NULL for no maximum)
    IN SaleYear INT                       -- Year filter for sale date (NULL for all years)
)
BEGIN
    -- setting/getting data
    SET @query = 'SELECT
        c.make AS Manufacturer,
        c.model AS Model,
        c.year AS Year,
        s.sale_price AS Sale_Price,
        s.sale_date AS Sale_Date,
        cu.first_name AS Customer_Name,
        cu.email AS Customer_Email,
        (SELECT ROUND(SUM(s2.sale_price), 2)
         FROM CarDealershipDataBase.Sales s2
         JOIN CarDealershipDataBase.Cars c2 ON s2.car_id = c2.car_id
         WHERE c2.make = c.make) AS Total_Revenue,
        (SELECT COUNT(*)
         FROM CarDealershipDataBase.Sales s3
         JOIN CarDealershipDataBase.Cars c3 ON s3.car_id = c3.car_id
         WHERE c3.make = c.make) AS Total_Cars_Sold
    FROM CarDealershipDataBase.Cars c
    JOIN CarDealershipDataBase.Sales s ON c.car_id = s.car_id
    JOIN CarDealershipDataBase.Customers cu ON s.customer_id = cu.customer_id
    WHERE 1 = 1';

    -- adding filters if input params are not null
    IF ManufacturerName IS NOT NULL THEN
  SET @query = CONCAT(@query, ' AND c.make = "', REPLACE(ManufacturerName, "'", "''"), '"');
    END IF;

    IF MinSalePrice IS NOT NULL THEN
        SET @query = CONCAT(@query, ' AND s.sale_price >= ', MinSalePrice);
    END IF;

    IF MaxSalePrice IS NOT NULL THEN
        SET @query = CONCAT(@query, ' AND s.sale_price <= ', MaxSalePrice);
    END IF;

    IF SaleYear IS NOT NULL THEN
        SET @query = CONCAT(@query, ' AND YEAR(s.sale_date) = ', SaleYear);
    END IF;

    -- default ordering by sale date desc
    SET @query = CONCAT(@query, ' ORDER BY s.sale_date DESC');

    -- execution
    PREPARE stmt FROM @query;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END$$
```

Output(s):

Example 1:

call GetFilteredSalesData('Tesla', 25000, 40000, null);

Manufacturer	Model	Year	Sale_Price	Sale_Date	Customer_Name	Customer_Email	Total_Revenue	Total_Cars_Sold
Tesla	Model S	2022	27000.00	2024-03-19	Mason	mason.hernandez@example.com	271000.00	5

Example 2:

call GetFilteredSalesData(null, null, null, null);

Manufacturer	Model	Year	Sale_Price	Sale_Date	Customer_Name	Customer_Email	Total_Revenue	Total_Cars_Sold
BMW	X5	2021	24000.00	2024-03-24	Liam	liam.perez@example.com	164500.00	4
Maserati	Quattroporte	2022	23000.00	2024-03-23	Evelyn	evelyn.gonzalez@example.com	23000.00	1
BMW	7 Series	2021	19500.00	2024-03-22	Isabella	isabella.lopez@example.com	164500.00	4
Tesla	Model 3	2022	21000.00	2024-03-21	Mia	mia.martinez@example.com	271000.00	5
Kia	Sorento	2018	25000.00	2024-03-20	Sophia	sophia.rodriguez@example.com	74000.00	3
Tesla	Model S	2022	27000.00	2024-03-19	Mason	mason.hernandez@example.com	271000.00	5
Mercedes	C-Class	2019	23500.00	2024-03-18	Henry	henry.martinez@example.com	131500.00	3
Jaguar	XJ	2019	18000.00	2024-03-17	Lucas	lucas.garcia@example.com	18000.00	1
Chevrolet	Tahoe	2021	22500.00	2024-03-16	Amelia	amelia.davis@example.com	133500.00	4
Chevrolet	Camaro	2021	20000.00	2024-03-15	Charlotte	charlotte.miller@example.com	133500.00	4
Audi	A4	2020	52000.00	2023-09-15	Liam	liam.perez@example.com	94000.00	2
BMW	X5	2021	59000.00	2023-07-21	Evelyn	evelyn.gonzalez@example.com	164500.00	4
Tesla	Model 3	2022	73000.00	2023-06-24	Aiden	aiden.johnson@example.com	271000.00	5
Honda	Civic	2019	18000.00	2023-05-25	Jessica	jessica.alba@example.com	47000.00	2
Honda	Accord	2019	29000.00	2023-04-12	Sophia	sophia.rodriguez@example.com	47000.00	2
Chevrolet	Camaro	2021	40000.00	2023-04-10	Robert	robert.downey@example.com	133500.00	4
Toyota	Corolla	2020	20000.00	2023-03-18	Michael	michael.jordan@example.com	54000.00	2
Nissan	Altima	2017	18000.00	2023-02-14	Lucas	lucas.garcia@example.com	18000.00	1
Tesla	Model 3	2022	70000.00	2023-01-10	James	james.williams@example.com	271000.00	5

Salesperson 6: Krish Sonani

Query 1 Identify Underperforming Salespersons

Objective: Find salespersons who sold cars with an average sale price below the overall average sale price.

```
SELECT sp.salesperson_id,
       CONCAT(sp.first_name, ' ', sp.last_name) AS salesperson_name,
       AVG(s.sale_price) AS avg_sale_price
FROM Sales s
JOIN Salespersons sp ON s.salesperson_id = sp.salesperson_id
GROUP BY sp.salesperson_id, sp.first_name, sp.last_name;
```

salesperson_id	salesperson_name	avg_sale_price
1	Alice Brown	23166.666667
2	Bob Johnson	39800.000000
3	David Clark	36125.000000
4	Sophia Martinez	41250.000000
5	James Moore	44500.000000
6	William Taylor	58000.000000
7	Ava Lee	21000.000000
8	Daniel Harris	26000.000000
9	Lucas Wilson	51000.000000
10	Emma White	29000.000000

Salesperson 6: Krish Sonani

Query2 Cars with Most Frequent Services

Objective: Retrieve the cars that have undergone the highest number of services, along with their service types and counts.

```sql
SELECT c.car_id,
       cm.make,
       cm.model,
       serv.service_type,
       COUNT(serv.service_id) AS service_count
FROM Services serv
JOIN Cars c ON serv.car_id = c.car_id
JOIN CarModels cm ON c.model_id = cm.model_id
GROUP BY c.car_id, cm.make, cm.model, serv.service_type
ORDER BY service_count DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap

car_id	make	model	service_type	service_count
1	Toyota	Corolla	Oil Change	2
2	Ford	Mustang	Tire Rotation	1
3	Tesla	Model S	Brake Inspection	1
4	Honda	Civic	Battery Replacement	1
5	Chevrolet	Camaro	Transmission Repair	1

Salesperson 6: Krish Sonani

Query3 Cars Performance Classification

Objective: Classify cars based on their stock status, mileage, and sales performance into categories like High Demand, Average Demand, and Low Demand

```sql
SELECT c.car_id,
       cm.make,
       cm.model,
       c.mileage,
       c.stock_status,
       COUNT(s.sale_id) AS total_sales,
       CASE
           WHEN c.stock_status = 'In Stock' AND c.mileage < 15000 AND COUNT(s.sale_id) > 5 THEN 'High Demand'
           WHEN c.stock_status = 'In Stock' AND c.mileage BETWEEN 15000 AND 30000 THEN 'Average Demand'
           WHEN c.stock_status = 'Out of Stock' AND COUNT(s.sale_id) = 0 THEN 'Low Demand'
           ELSE 'Moderate Demand'
       END AS performance_category
FROM Cars c
JOIN CarModels cm ON c.model_id = cm.model_id
LEFT JOIN Sales s ON c.car_id = s.car_id
GROUP BY c.car_id, cm.make, cm.model, c.mileage, c.stock_status
ORDER BY performance_category DESC, total_sales DESC;
```

car_id	make	model	mileage	stock_status	total_sales	performance_category
9	Tesla	Model 3	1500	In Stock	3	Moderate Demand
6	BMW	X5	8000	In Stock	3	Moderate Demand
8	Mercedes	C-Class	12000	Sold	3	Moderate Demand
15	Kia	Sorento	11500	Sold	3	Moderate Demand
3	Tesla	Model S	1200	In Stock	2	Moderate Demand
5	Chevrolet	Camaro	7000	Sold	2	Moderate Demand
7	Audi	A4	9000	In Stock	2	Moderate Demand
12	Chevrolet	Tahoe	8000	In Stock	2	Moderate Demand
2	Ford	Mustang	5000	Sold	1	Moderate Demand
4	Honda	Civic	12000	In Stock	1	Moderate Demand
17	BMW	7 Series	3000	In Stock	1	Moderate Demand
10	Toyota	Highlan...	18000	Sold	1	Moderate Demand
11	Ford	Escape	7500	In Stock	1	Moderate Demand
13	Honda	Accord	10000	Sold	1	Moderate Demand
14	Nissan	Altima	6500	In Stock	1	Moderate Demand
24	Bentley	Contine...	1000	In Stock	1	Moderate Demand
21	Jaguar	XJ	15000	Out of Stock	1	Moderate Demand
16	Mercede...	S-Class	5000	In Stock	0	Moderate Demand
69	Nissan	Altima	900	In Stock	0	Moderate Demand
18	Audi	A8	2500	In Stock	0	Moderate Demand
19	Porsche	Panamera	4000	In Stock	0	Moderate Demand
20	Lexus	LS	7000	In Stock	0	Moderate Demand
22	Land Rover	Range ...	2000	In Stock	0	Moderate Demand
23	Maserati	Quattro...	6000	In Stock	0	Moderate Demand
70	Dodge	Journey	1100	In Stock	0	Moderate Demand

Salesperson 7: Ishan Waghmare

Query 1: List All Cars Priced Above $50,000, Ordered by Price.

Objective: Highlight premium vehicles, allow dealers to focus on high-value sales opportunities & customer targeting.

```
555    #List All Cars Priced Above $50,000, Ordered by Price
556    SELECT car_id, model_id, price, color
557    FROM Cars
558    WHERE price > 50000
559    ORDER BY price DESC;
560
```

Result Grid | Filter Rows: | Edit: | Export/Im

car_id	model_id	price	color
25	25	250000.00	Blue
19	19	120000.00	Red
24	24	120000.00	Black
16	16	110000.00	Black
23	23	110000.00	White
18	18	105000.00	Silver
17	17	100000.00	White
22	22	95000.00	Gray
20	20	90000.00	Blue
21	21	85000.00	Green
3	3	80000.00	Black

Salesperson 7: Ishan Waghmare

Query 2: Find the Number of Cars Sold by Each Salesperson, Ordered by Total Cars Sold.

Objective: Access individual salesperson performance, informing incentives and resources allocation.

```
561    #Find the Number of Cars Sold by Each Salesperson, Ordered by Total Cars Sold
562 •  SELECT salesperson_id, COUNT(sale_id) AS CarsSold
563    FROM Sales
564    GROUP BY salesperson_id
565    ORDER BY CarsSold DESC;
566
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

salesperson_id	CarsSold
1	6
2	5
3	4
4	4
5	4
7	2
6	2
8	1
9	1
10	1

Salesperson 7: Ishan Waghmare

Query 3: List All Cars Sold by a Specific Salesperson (e.g., ID = 3).

Objective: Track detailed sales history for individual employee, aids in performance reviews & customer follow-ups.

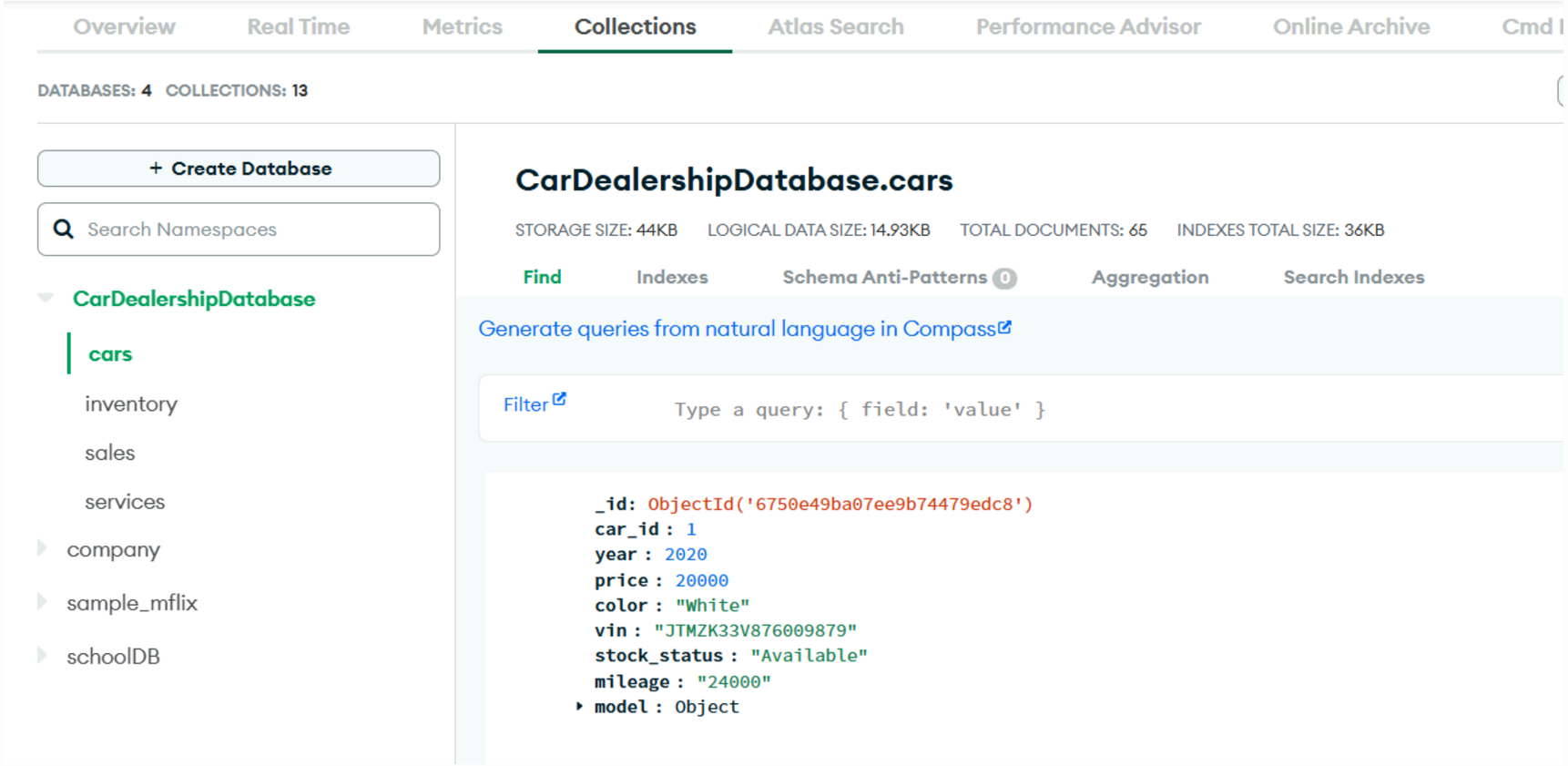
```
566
567     #List All Cars Sold by a Specific Salesperson
568 •   SELECT s.sale_id, c.car_id, c.vin, s.sale_price, s.sale_date
569     FROM Sales s
570     JOIN Cars c ON s.car_id = c.car_id
571     WHERE s.salesperson_id = 3
572     ORDER BY s.sale_date DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sale_id	car_id	vin	sale_price	sale_date
27	9	5YJ3E1EA1JF123456	21000.00	2024-03-21
22	12	1GNFK13027J123456	22500.00	2024-03-16
16	6	WBAXH5C59F0K12345	59000.00	2023-07-21
6	7	WAUZZZF48HA123456	42000.00	2022-06-15

NoSQL Database

MongoDB is a non-relational database platform for handling unstructured, semi-structured, or structured data. They provide high flexibility, scalability, and performance.



Add Ons (Graphical User Interface)

🚘 This GUI simplifies managing an extensive car dealership inventory interactively 🚘

Python Tools

- Used the ‘Tkinter’ library which is a Python standard library for creating GUIs, so no external installations were required!
  - basic building blocks like buttons, labels, entry fields, menus, and text boxes
  - supports styling and animations!

Key Features/Interactions of GUI

1. Car Inventory Display:
   - Displays a list of cars in a scrollable table (Treeview).
   - Columns include ID, Make, Model, Year, Price, Color, Fuel Type, VIN, Status, and Mileage.
2. Add New Cars:
   - Opens a form to add a new car with all necessary details.
   - Required Fields: Car ID, Make, Model, Year, Price, Color, Fuel Type, VIN, Status, Mileage.
3. Update Existing Cars:
   - Allows updating the selected car's details.
   - Prefills the current car information for easier editing.
4. Delete Existing Cars:
   - Lets the user delete an existing car and its details.
5. Interactive UI:
   - The Treeview (table) and buttons are fully resizable and adjust to window sizing.

ID	Make	Model	Year	Price	Color	Fuel	VIN	Status	Mileage
1	Toyota	Corolla	2020	20000.0	White	Gasoline	JTMZK33V8760	Available	15000
2	Ford	Mustang	2021	35000.0	Red	Gasoline	1ZVBP8CF6B51	Sold	5000
3	Tesla	Model S	2022	80000.0	Black	Electric	5YJSA1E25FF10	Available	1200
4	Honda	Civic	2019	18000.0	Blue	Gasoline	19XFC2F59JE21	Available	12000
5	Chevrolet	Camaro	2021	40000.0	Yellow	Gasoline	1G1F51R63J010	Sold	7000
6	Lexus	LS	2020	90000.0	Blue	Hybrid	5LEXUSXXJXPN	Available	9500
7	BMW	3 series	2023	70000.0	Dravit Grey	Hybrid	WBAXH5C59F0	Available	8000
8	Mercedes-Benz	S-Class	2022	110000.0	White	Gasoline	1MBEJXXJXPN1	Available	13000
9	Hyundai	Accent	2022	17500.0	Silver	Gasoline	3HYUNDAIACX	Available	35000
10	Land Rover	Range Rover Sp	2024	140000.0	Green	Gasoline	8LANDROVERX	Sold	3000

Add Car | Update Car | Delete Car

Innovative Components – Implemented a car animation!

- Utilizing an element from the ‘Tkinter’ library called Canvas which allows for graphical animations.

Created an animation logic resembling the movement of a car on a road


DEMO

QUESTIONS?

Feel free to reach out to us!!!

THANK YOU FOR YOUR TIME!