Transaction(s):
```
conn.start_transaction(isolation_level='READ COMMITTED')

  # --- Advanced query 1: Aggregate rentals per garage ---
  cur.execute("""
  SELECT g.GarageId, COUNT(r.RentalId) AS total_rentals
  FROM Garages g
  LEFT JOIN Rentals r ON g.GarageId = r.GarageId
  JOIN GarageLocation gl ON g.Latitude = gl.Latitude AND g.Longitude = gl.Longitude
  WHERE gl.City = %s
  GROUP BY g.GarageId
  """, (city,))

  # --- Advanced query 2: Rental with highest total cost for each garage ---
  cur.execute("""
  SELECT g.GarageId, r.RentalId, r.TotalCost, r.StartTime, r.EndTime
  FROM Rentals r
  JOIN CarData c ON r.CarId = c.CarId
  JOIN Garages g ON c.GarageId = g.GarageId
  JOIN GarageLocation gl ON g.Latitude = gl.Latitude AND g.Longitude = gl.Longitude
  WHERE gl.City = %s AND r.TotalCost = (
    SELECT MAX(r2.TotalCost)
    FROM Rentals r2
    JOIN CarData c2 ON r2.CarId = c2.CarId
    WHERE c2.GarageId = g.GarageId
  )
  ORDER BY g.GarageId
  """, (city,))
```

Trigger: Trigger to maintain real-time Garage profits after rental transactions.

```
DELIMITER $$

CREATE TRIGGER after_rental_insert
AFTER INSERT ON Rentals
FOR EACH ROW
BEGIN
```

```
  UPDATE Garages
  SET Profit = COALESCE(Profit, 0) + NEW.TotalCost
  WHERE GarageId = NEW.GarageId;
END$$

DELIMITER ;
```

Stored procedure(s):

```
CREATE DEFINER=`root`@`%` PROCEDURE `GetCustomerRentalSummaryById`(IN
p_CustomerId INT)
BEGIN
  DECLARE customerTotalSpending DECIMAL(10,2);

  -- customer's total spending
  SELECT COALESCE(SUM(r.TotalCost), 0)
  INTO customerTotalSpending
  FROM Rentals r
  WHERE r.CustomerId = p_CustomerId;

  -- rank
  SELECT main.CustomerId,
    main.CustomerName,
    main.TotalSpending,
    main.AverageRating,
    CASE WHEN main.NumberOfCustomersWithHigherSpending IS NULL THEN 1
      ELSE main.NumberOfCustomersWithHigherSpending + 1
    END AS CustomerSpendingRank
  FROM (
    SELECT c.CustomerId,
      c.Name AS CustomerName,
      COALESCE(SUM(r.TotalCost), 0) AS TotalSpending,
      COALESCE(AVG(cs.Rating), 0) AS AverageRating,
      (
        SELECT COUNT(DISTINCT c2.CustomerId)
        FROM Customers c2
        JOIN Rentals r2 ON c2.CustomerId = r2.CustomerId
        GROUP BY c2.CustomerId
        HAVING SUM(r2.TotalCost) > customerTotalSpending
      ) AS NumberOfCustomersWithHigherSpending
    FROM Customers c
    LEFT JOIN Rentals r ON c.CustomerId = r.CustomerId
    LEFT JOIN CustomerSatisfaction cs ON r.RentalId = cs.RentalId
    WHERE c.CustomerId = p_CustomerId
    GROUP BY c.CustomerId, c.Name
  ) AS main;
END
```