Unit 3 SQL Aggregates and Groupings

**Corey Crooks**

**Purdue University Global**

**IT350 – Advanced Database Concepts**

**Jason Johnson**

**February 22nd, 2023**

**Problem 1:** Establish a database view called TotalBikesSoldView that lists each bike name and associated total quantity sold. Then query the view to show the top ten bikes sold. Present the output in descending order based on total bikes sold.

USE BikeStores

CREATE VIEW TotalBikesSoldView AS

SELECT product\_name, SUM(quantity) AS "Total Sold"

FROM Production.Products PR

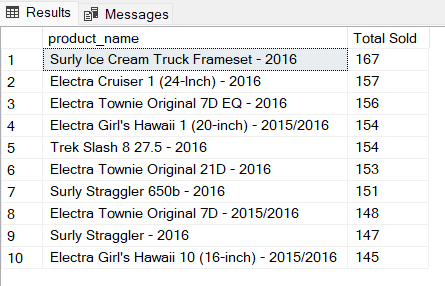
INNER JOIN Sales.Order\_Items OI

ON PR.product\_id = OI.product\_id

GROUP BY product\_name;

SELECT TOP 10 \* FROM TotalBikesSoldView

ORDER BY "Total Sold" DESC;



**Problem 2:** Establish a database view called **MaximumBrandBikePriceView** that lists each bike brand and associated maximum bike price. Then query the view and present the output in ascending order based on the bike brand name.

USE BikeStores

CREATE VIEW MaximumBrandBikePriceView AS

SELECT brand\_name, MAX(list\_price) AS MaxPrice

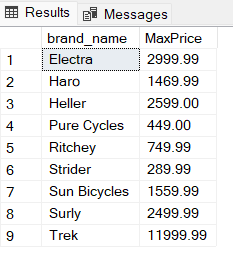
FROM Production.Products PR INNER JOIN Production.Brands BR

ON PR.brand\_id = BR.brand\_id

GROUP BY brand\_name;

SELECT \* FROM MaximumBrandBikePriceView

ORDER BY brand\_name;



**Problem 3:** Establish a database view called **AverageBrandBikePriceView** that lists each bike brand name and associated average bike list price and list price standard deviation. Then query the view and present the output in ascending order based on the bike brand name.

USE BikeStores

CREATE VIEW AverageBrandBikePriceView AS

SELECT brand\_name, AVG(list\_price) AS AvgPrice,

STDEV(list\_price) AS StdDevPrice

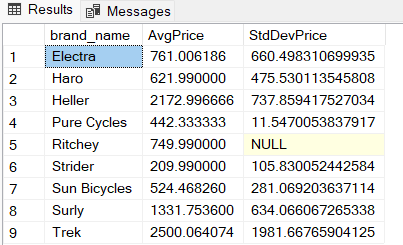
FROM Production.Products PR INNER JOIN Production.Brands BR

ON PR.brand\_id = BR.brand\_id

GROUP BY brand\_name;

SELECT \* FROM AverageBrandBikePriceView

ORDER BY brand\_name;



**Problem 4:** Establish a database view called **NumberOfCustomersPerStateView** that lists the number of customers in each state. The view needs to show the state name along with the customer amount. Then query the view and present the output in ascending order based on state.

USE BikeStores

CREATE VIEW NumberOfCustomersPerStateView AS

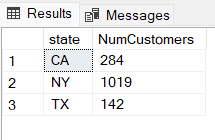
SELECT state, COUNT(\*) NumCustomers

FROM Sales.customers

GROUP BY state;

SELECT \* FROM NumberOfCustomersPerStateView

ORDER BY state;



**Problem 5:** Establish a database view called **TotalBikesSoldPerStateView** that shows the cumulative bike sales per state. The view needs to show the state name along with the bike sales quantity. Then query the view and present the output in ascending order based on state.

USE BikeStores

CREATE VIEW TotalBikesSoldPerStateView AS

SELECT state, SUM(quantity) AS NumBikesSold

FROM Sales.customers C INNER JOIN Sales.orders O

ON C.customer\_id = O.order\_id

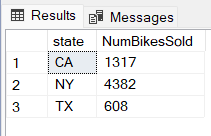
INNER JOIN Sales.order\_items OI

ON O.order\_id = OI.order\_id

GROUP BY state;

SELECT \* FROM TotalBikesSoldPerStateView

ORDER BY state;



**Problem 6:**Create a stored procedure that lists the stock quantity at each store for a specified bike brand. The output must show the store name and stock quantity value. The results need to be presented in ascending order based on the store name. The skeleton of the stored procedure is provided below.

USE BikeStores

CREATE PROCEDURE GetStoreBrandStockQuantity

@BrandName VARCHAR(255)

AS

SELECT store\_name, SUM(quantity) AS StockAmt

FROM Sales.stores ST INNER JOIN Production.stocks SK

ON ST.store\_id = SK.store\_id

INNER JOIN Production.products PR

ON SK.product\_id = PR.product\_id

INNER JOIN Production.Brands BR

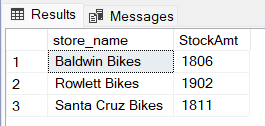
ON PR.brand\_id = BR.brand\_id

WHERE brand\_name = @BrandName

GROUP BY store\_name

ORDER BY store\_name;

EXEC GetStoreBrandStockQuantity @BrandName='Trek';



**Problem 7:** Create a stored procedure that lists the average, standard deviation, maximum, and minimum list prices for a specified bike brand. The output must show the brand name along with the aggregated results. The results need to be presented in ascending order based on the store name. The skeleton of the stored procedure is provided below.

USE BikeStores

CREATE PROCEDURE GetBrandListPriceStatistics

@BrandName VARCHAR(255)

AS

SELECT brand\_name,

AVG (list\_price) AS PriceAvg,

STDEV(list\_price) AS PriceStdDev,

MAX (list\_price) AS PriceMax,

MIN(list\_price) AS PriceMin

FROM Production.products PR

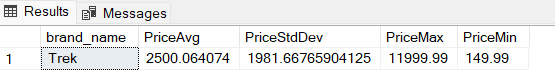
INNER JOIN Production.brands BR

ON PR.brand\_id = BR.brand\_id

WHERE brand\_name = @BrandName

GROUP BY brand\_name;

EXEC GetBrandListPriceStatistics @BrandName='Trek';



**Problem 8:** Create a stored procedure that lists the bike categories with minimum list prices below a specified value. The output must show the bike category and associated minimum list price. The results need to be presented in ascending order based on the bike category. The skeleton of the stored procedure is provided below. (Hint: The HAVING clause needs to be used.)

USE BikeStores

CREATE PROCEDURE GetBikeCategoriesWithMinPriceBelowValue

@Price DECIMAL(10,2)

AS

SELECT category\_name, MIN(list\_price) AS MinListPrice

FROM Production.Products PR INNER JOIN Production.Categories CA

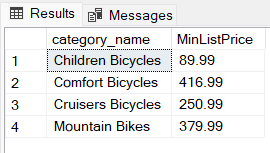
ON PR.category\_id = CA.category\_id

GROUP BY category\_name

HAVING MIN(list\_price) <= @Price

ORDER BY category\_name;

EXEC GetBikeCategoriesWithMinPriceBelowValue @Price=500;



**Problem 9:** Create a stored procedure that lists the stock quantity for each bike category at a specified store. The output must show the bike category and stock quantity value. The results need to be presented in ascending order based on the bike category. The skeleton of the stored procedure is provided below.

USE BikeStores

CREATE PROCEDURE GetCategoryStockQuantityForStore

@StoreName VARCHAR(255)

AS

SELECT category\_name, SUM(quantity) AS NumBikes

FROM Production.categories CT INNER JOIN Production.products PR

ON CT.category\_id = PR.category\_id

INNER JOIN Production.stocks SK

ON PR.product\_id = SK.product\_id

INNER JOIN Sales.stores ST

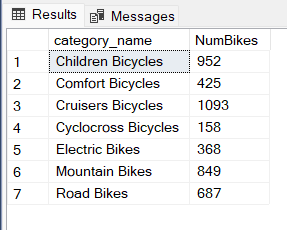
ON SK.store\_id = ST.store\_id

WHERE ST.store\_name = @StoreName

GROUP BY category\_name

ORDER BY category\_name;

EXEC GetCategoryStockQuantityForStore @StoreName='Santa Cruz Bikes';



**Problem 10:** Create a stored procedure that presents the sales discount average and standard deviation for bike brands at or above a specified discount value. The output must show the brand names in alphabetical order along with the associated statistics. The skeleton of the stored procedure is provided below. (Hint: The HAVING clause needs to be used.)

USE BikeStores

CREATE PROCEDURE GetBrandAverageSalesDiscountAboveValue

@Discount DECIMAL(4,2)

AS

SELECT brand\_name,

AVG(discount) AS DiscAvg,

STDEV(discount) AS DiscStdDev

FROM production.brands BR INNER JOIN production.products PR

ON BR.brand\_id = PR.brand\_id

INNER JOIN Sales.order\_items OI

ON PR.product\_id = OI.product\_id

GROUP BY brand\_name

HAVING AVG(discount) >= @Discount

ORDER BY brand\_name;

EXEC GetBrandAverageSalesDiscountAboveValue @Discount=0.11;

