Data Wrangling and R

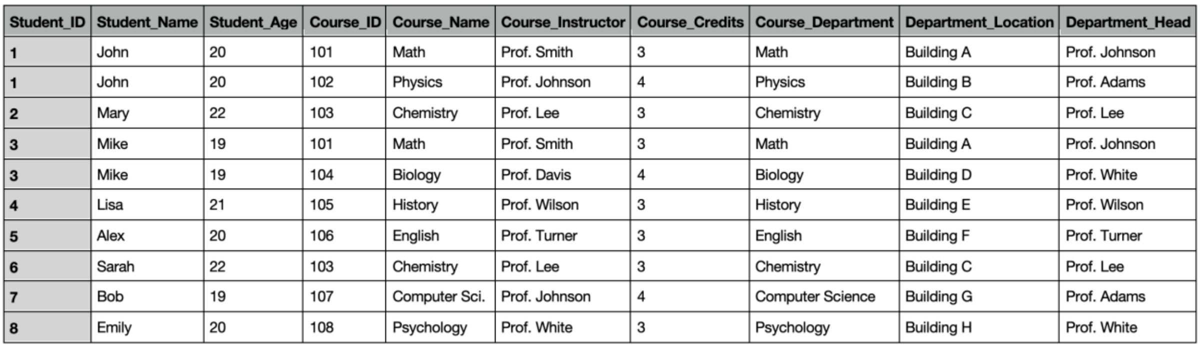
Overview

Over the past few weeks, This report have a general understanding of designing databases and how to use SQL to retrieve information from business data.

Table of Contents

Part 1: Database design

Analysis of University Students and Courses: Key Insights and Findings



Student_ID	Student_Name	Student_Age	Course_ID	Course_Name	Course_Instructor	Course_Credits	Course_Department	Department_Location	Department_Head
1	John	20	101	Math	Prof. Smith	3	Math	Building A	Prof. Johnson
1	John	20	102	Physics	Prof. Johnson	4	Physics	Building B	Prof. Adams
2	Mary	22	103	Chemistry	Prof. Lee	3	Chemistry	Building C	Prof. Lee
3	Mike	19	101	Math	Prof. Smith	3	Math	Building A	Prof. Johnson
3	Mike	19	104	Biology	Prof. Davis	4	Biology	Building D	Prof. White
4	Lisa	21	105	History	Prof. Wilson	3	History	Building E	Prof. Wilson
5	Alex	20	106	English	Prof. Turner	3	English	Building F	Prof. Turner
6	Sarah	22	103	Chemistry	Prof. Lee	3	Chemistry	Building C	Prof. Lee
7	Bob	19	107	Computer Sci.	Prof. Johnson	4	Computer Science	Building G	Prof. Adams
8	Emily	20	108	Psychology	Prof. White	3	Psychology	Building H	Prof. White

1.1. Evaluating Normal Forms: 1NF, 2NF, and 3NF Analysis with Anomalies in Data Management

The table satisfies the first normal form (1NF) because:

- The table does not have any attributes that hold multiple values.

- It hold only atomic values (no multi-valued attributes).

The table does not satisfy the second normal form (2NF) because both Student_ID and Course_ID are primary keys and Course_Credits is a non-key attribute then Course_Credits (non-key attribute) is partially dependent on Course_ID, not the whole primary key. This violates the 2NF.

The table does not satisfy the third normal form (3NF) because it does not satisfy the second normal form (2NF).

Examples:

- Insertion anomaly: Because the primary keys are Student_ID and Course_ID, we cannot add a new Course_Name before having a student enrolled in it.

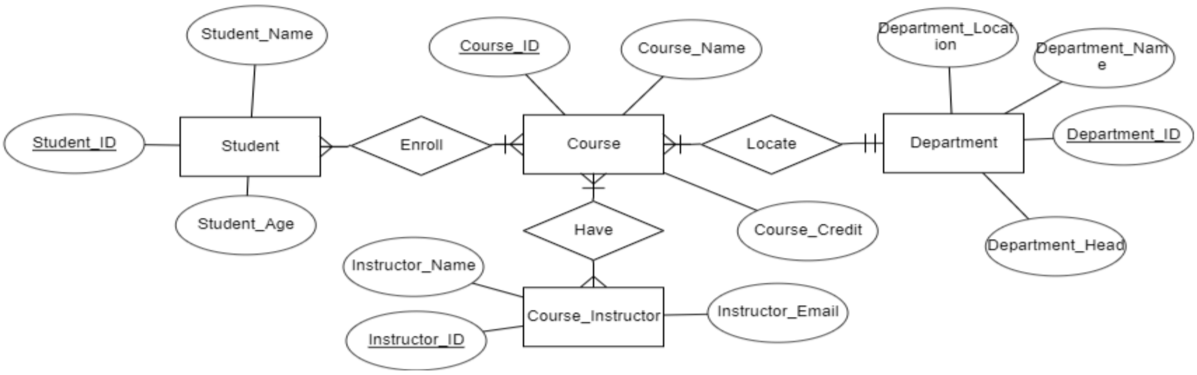
- Delection anomaly: deleting the row have Student_Name "Lisa", the information related to Course_Name "History" included Course_Instructor, Course_ Department and Department_Location will be deleted.

- Update anomaly: a change of Biology Department_Head "Prof.White" requires multiple updates because he is also the course instructor of Psychology Course and the Department Head of Psychology Department.

1.2. Redesigning for 3NF: ERD Creation, Relational Schema, and SQL Table Generation

Answer:

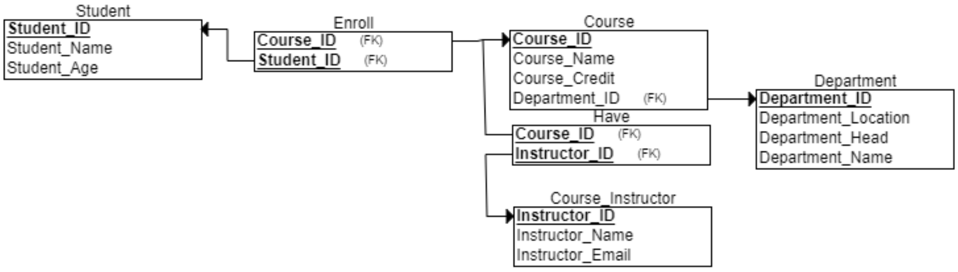
1.2.1. ERD Diagram



Constraints imposed by the cardinality of the relationships:

1. A student must enroll at least one course to appear in the system and may enroll many courses.
2. A course can have multiple students (many students).
3. A course must be located in one (and only one) department.
4. A department must have at least one course and may have many courses.
5. A course may have many course instructors.
6. A course instructor must attend at least one course and may attend many courses.

1.2.2. Converted relational schema



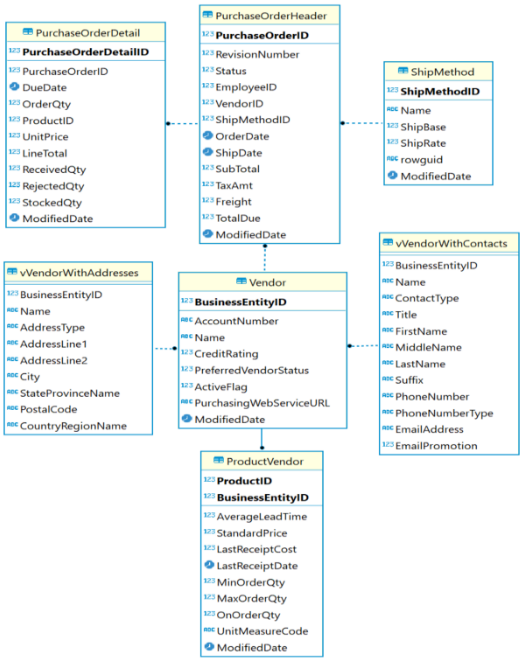
1.2.3. Generated SQL statements

```
CREATE TABLE Student

(

 Student_ID INT NOT NULL,

 Student_Name VARCHAR(250) NOT NULL,

 Student_Age NUMERIC(2) NOT NULL,

 PRIMARY KEY (Student_ID)

);


CREATE TABLE Department

(

 Department_Location VARCHAR(250) NOT NULL,

 Department_Head VARCHAR(250) NOT NULL,

 Department_ID INT NOT NULL,

 Department_Name VARCHAR(40) NOT NULL,
  PRIMARY KEY (Department_ID)
);

CREATE TABLE Course_Instructor
(
  Instructor_Name VARCHAR(250) NOT NULL,
  Instructor_Email VARCHAR(250) NOT NULL,
  Instructor_ID INT NOT NULL,
  PRIMARY KEY (Instructor_ID)
);

CREATE TABLE Course
(
  Course_ID INT NOT NULL,
  Course_Name VARCHAR(250) NOT NULL,
  Course_Credit NUMERIC(2) NOT NULL,
  Department_ID INT NOT NULL,
  PRIMARY KEY (Course_ID),
  FOREIGN KEY (Department_ID) REFERENCES Department(Department_ID)
);

CREATE TABLE Enroll
(
  Course_ID INT NOT NULL,
  Student_ID INT NOT NULL,
  PRIMARY KEY (Course_ID, Student_ID),
  FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID),
  FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
);

CREATE TABLE Have
(
  Course_ID INT NOT NULL,
  Instructor_ID INT NOT NULL,
  PRIMARY KEY (Course_ID, Instructor_ID),
  FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID),
  FOREIGN KEY (Instructor_ID) REFERENCES Course_Instructor(Instructor_ID)
);
```

Part 2: Database retrieval using SQL

2.1. Defining Table Relationships: Constraints, Primary Keys, and Foreign Keys in Database Design

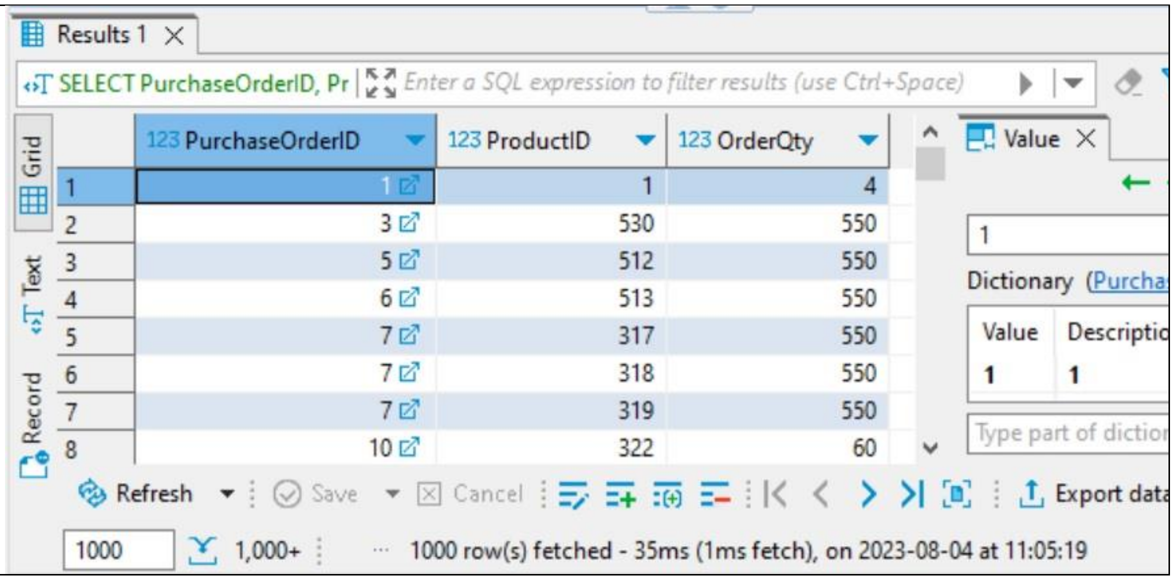
- Purchase Order Detail and Purchase Order Header:
  - One Purchase Order Detail can specify only one Purchase Order.
  - One Purchase Order can appear in multiple Purcase Order Detail.
- Purchase Order Header and Ship Method:
  - One Purchase Order can specify only one Ship Method.
  - One Ship Method can appear in multiple Purchase Order.
- Vendor and Purchase Order Header
  - One Purchase Order can specify only one Vendor.
  - One Vendor can appear in multiple Purchase Order.
- Vendor and Product Vendor
  - One Product can have multiple Vendors.
  - One Vendor can appear in multiple Products.



2.2. Using SQL for Business Insights: Answering Key Questions to Support Managerial Decisions.

2.2.1. Display the orders that have an order quantity greater than 3.

Code	SELECT pod.PurchaseOrderID, pod.ProductID , pod.OrderQty
	FROM Purchasing.PurchaseOrderDetail pod
	WHERE pod.OrderQty > 3

Returned table	

SELECT PurchaseOrderID, Pr

	PurchaseOrderID	ProductID	OrderQty
1	1	1	4
2	3	530	550
3	5	512	550
4	6	513	550
5	7	317	550
6	7	318	550
7	7	319	550
8	10	322	60

Value: 1

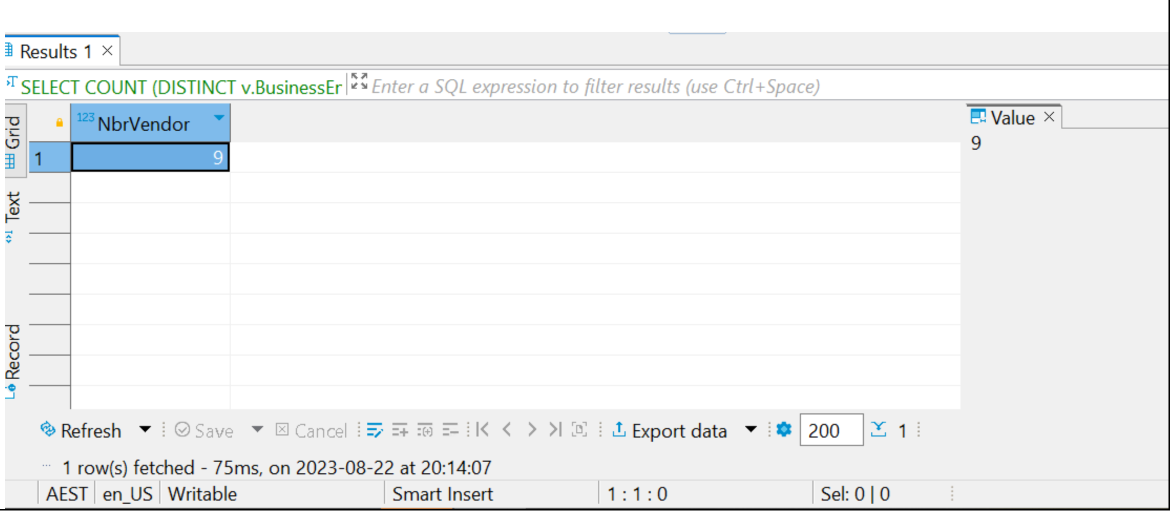
Dictionary (Purcha

Value	Descriptio
1	1

1000 row(s) fetched - 35ms (1ms fetch), on 2023-08-04 at 11:05:19

2.2.2. Querying Vendor Orders: Counting Vendors with Credit Ratings Above 2

Code	SELECT COUNT (DISTINCT v.BusinessEntityID) as NbrVendor
	FROM Purchasing.Vendor v
	JOIN Purchasing.PurchaseOrderHeader poh
	on v.BusinessEntityID = poh.VendorID
	WHERE v.CreditRating > 2

Returned table	

SELECT COUNT (DISTINCT v.BusinessEr

	NbrVendor
1	9

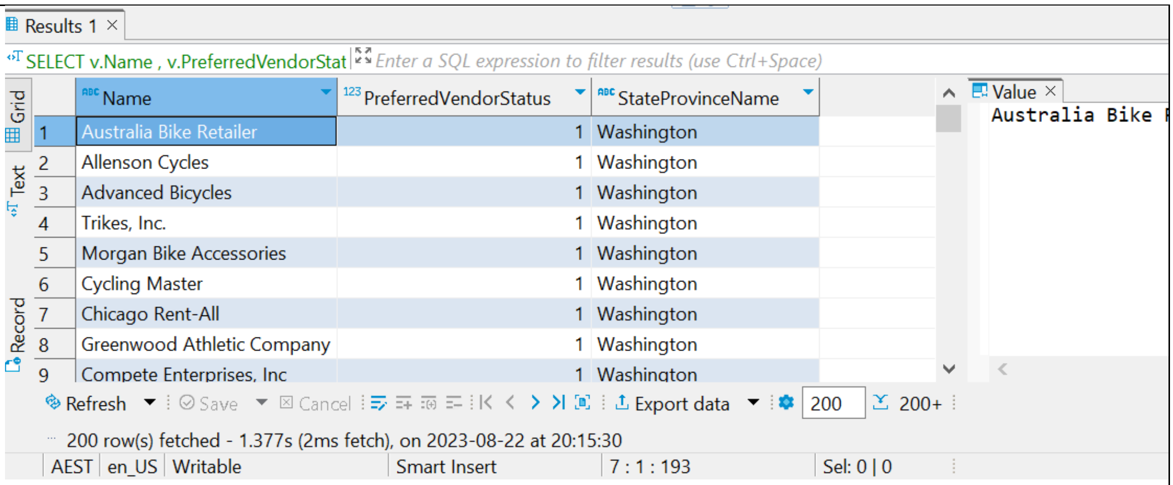
Value: 9

1 row(s) fetched - 75ms, on 2023-08-22 at 20:14:07

AEST | en_US | Writable | Smart Insert | 1 : 1 : 0 | Sel: 0 | 0

2.2.3. Display names of vendors and their preferred status of vendors who live in Washington province.

Code	SELECT v.Name , v.PreferredVendorStatus, vvwa.StateProvinceName
	FROM Purchasing.Vendor v, Purchasing.vVendorWithAddresses vvwa
	WHERE vvwa.StateProvinceName = 'Washington'

Returned table	

Results 1

SELECT v.Name , v.PreferredVendorStat | Enter a SQL expression to filter results (use Ctrl+Space)

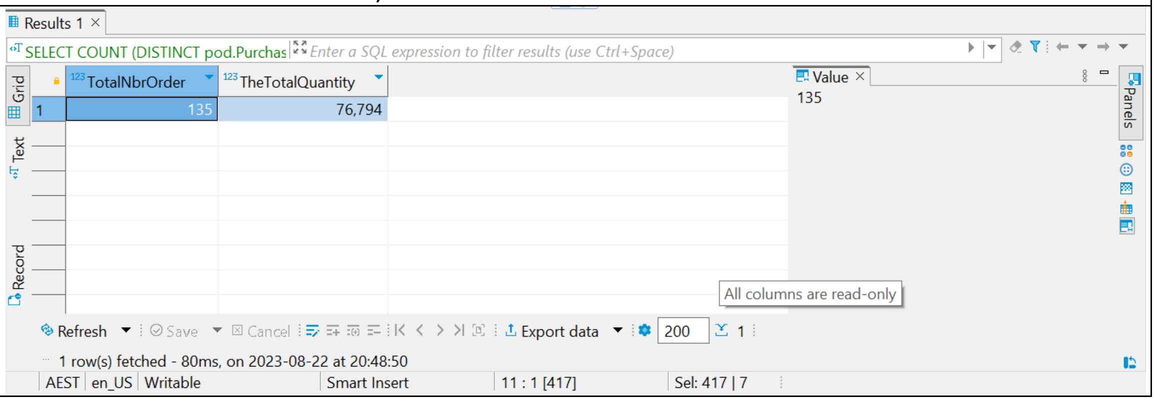
	Name	PreferredVendorStatus	StateProvinceName
1	Australia Bike Retailer	1	Washington
2	Allenson Cycles	1	Washington
3	Advanced Bicycles	1	Washington
4	Trikes, Inc.	1	Washington
5	Morgan Bike Accessories	1	Washington
6	Cycling Master	1	Washington
7	Chicago Rent-All	1	Washington
8	Greenwood Athletic Company	1	Washington
9	Compete Enterprises, Inc	1	Washington

Value: Australia Bike R

200 row(s) fetched - 1.377s (2ms fetch), on 2023-08-22 at 20:15:30

AEST | en_US | Writable | Smart Insert | 7 : 1 : 193 | Sel: 0 | 0

2.2.4. Distinct 2012 Orders Shipped by 'Truck' or 'Cargo Ship': Count and Total Quantity

Code	**SELECT COUNT** (**DISTINCT** pod.PurchaseOrderID) **as** TotalNbrOrder, **SUM**(pod.OrderQty) **as** TheTotalQuantity **FROM** Purchasing.PurchaseOrderHeader poh **JOIN** Purchasing.ShipMethod sm **on** sm.ShipMethodID = poh.ShipMethodID **JOIN** Purchasing.PurchaseOrderDetail pod **on** pod.PurchaseOrderID = poh.PurchaseOrderID **WHERE** (sm.Name **LIKE** '%truck%' **or** sm.Name **LIKE** '%cargo%') **AND** (poh.OrderDate **BETWEEN** '2012-01-01' **AND** '2012-12-31')
Returned table	

Results 1

SELECT COUNT (DISTINCT pod.Purchas | Enter a SQL expression to filter results (use Ctrl+Space)

	TotalNbrOrder	TheTotalQuantity
1	135	76,794

Value: 135

All columns are read-only

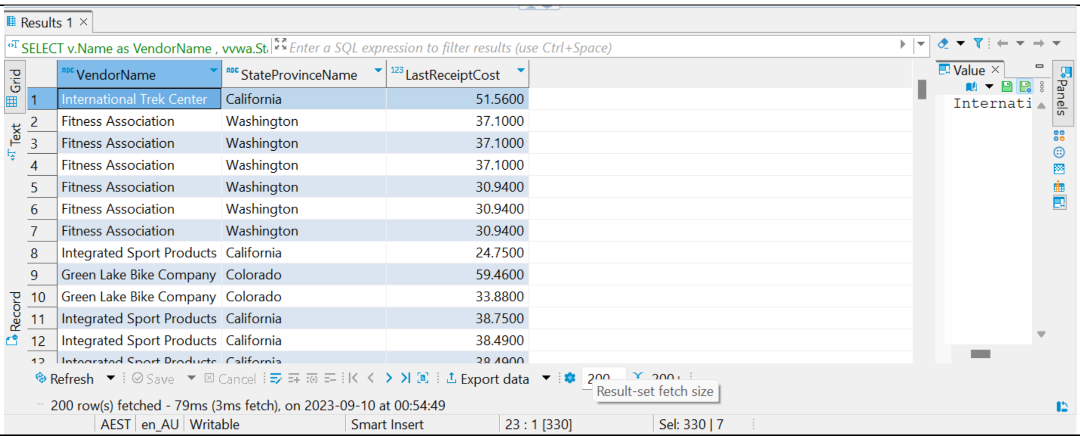
1 row(s) fetched - 80ms, on 2023-08-22 at 20:48:50

AEST | en_US | Writable | Smart Insert | 11 : 1 [417] | Sel: 417 | 7

2.2.5. Display vendors name, state and its latest receipt cost in descending order of both standard price and average lead times.

Code	**SELECT** v.Name **as** VendorName , vvwa.StateProvinceName , pv.LastReceiptCost **FROM** Purchasing.Vendor v **JOIN** Purchasing.vVendorWithAddresses vvwa **on** v.BusinessEntityID = vvwa.BusinessEntityID **JOIN** Purchasing.ProductVendor pv **on** pv.BusinessEntityID = v.BusinessEntityID **ORDER BY** pv.AverageLeadTime **DESC**, pv.StandardPrice **DESC**

Returned table



Results 1

SELECT v.Name as VendorName , vvwa.St

	VendorName	StateProvinceName	LastReceiptCost
1	International Trek Center	California	51.5600
2	Fitness Association	Washington	37.1000
3	Fitness Association	Washington	37.1000
4	Fitness Association	Washington	37.1000
5	Fitness Association	Washington	30.9400
6	Fitness Association	Washington	30.9400
7	Fitness Association	Washington	30.9400
8	Integrated Sport Products	California	24.7500
9	Green Lake Bike Company	Colorado	59.4600
10	Green Lake Bike Company	Colorado	33.8800
11	Integrated Sport Products	California	38.7500
12	Integrated Sport Products	California	38.4900
13	Integrated Sport Products	California	38.4900

200 row(s) fetched - 79ms (3ms fetch), on 2023-09-10 at 00:54:49

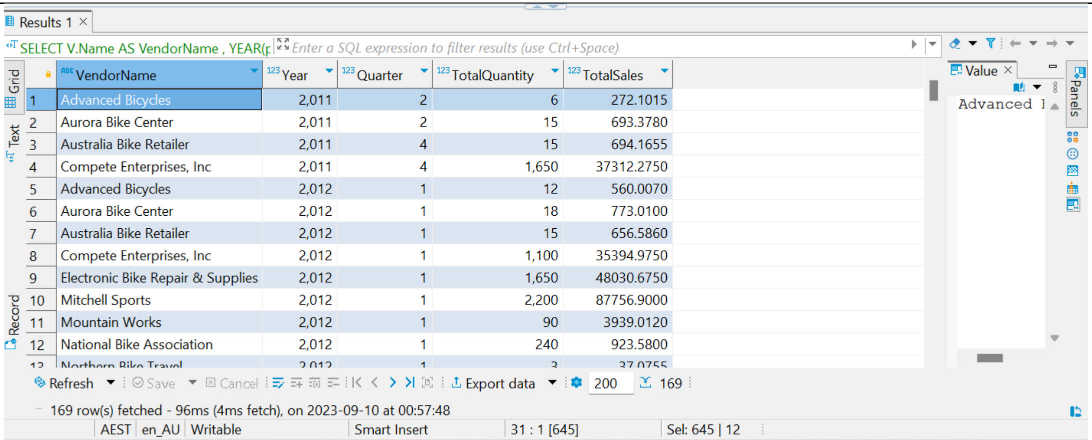
AEST | en_AU | Writable | Smart Insert | 23 : 1 [330] | Sel: 330 | 7

2.2.6. Calculate the quarter sales of vendors in Washington province.

Code

```
SELECT V.Name AS VendorName , YEAR(poh.OrderDate) AS 'Year',
DATEPART(QUARTER,poh.OrderDate) AS 'Quarter',
SUM(pod.OrderQty) as TotalQuantity, SUM(pod.LineTotal) AS TotalSales
FROM Purchasing.PurchaseOrderHeader poh
JOIN Purchasing.PurchaseOrderDetail pod
ON pod.PurchaseOrderID = poh.PurchaseOrderID
JOIN Purchasing.Vendor v
ON v.BusinessEntityID = poh.VendorID
JOIN Purchasing.vVendorWithAddresses vvwa
ON vvwa.BusinessEntityID = v.BusinessEntityID
WHERE vvwa.StateProvinceName ='Washington'
GROUP BY v.Name, YEAR(poh.OrderDate), DATEPART(QUARTER,poh.Orderdate)
ORDER BY YEAR(poh.Orderdate), DATEPART (QUARTER,poh.Orderdate)
```

Returned table



Results 1

SELECT V.Name AS VendorName , YEAR(p

	VendorName	Year	Quarter	TotalQuantity	TotalSales
1	Advanced Bicycles	2,011	2	6	272.1015
2	Aurora Bike Center	2,011	2	15	693.3780
3	Australia Bike Retailer	2,011	4	15	694.1655
4	Compete Enterprises, Inc	2,011	4	1,650	37312.2750
5	Advanced Bicycles	2,012	1	12	560.0070
6	Aurora Bike Center	2,012	1	18	773.0100
7	Australia Bike Retailer	2,012	1	15	656.5860
8	Compete Enterprises, Inc	2,012	1	1,100	35394.9750
9	Electronic Bike Repair & Supplies	2,012	1	1,650	48030.6750
10	Mitchell Sports	2,012	1	2,200	87756.9000
11	Mountain Works	2,012	1	90	3939.0120
12	National Bike Association	2,012	1	240	923.5800
13	Northern Bike Travel	2,012	1	3	37.0755

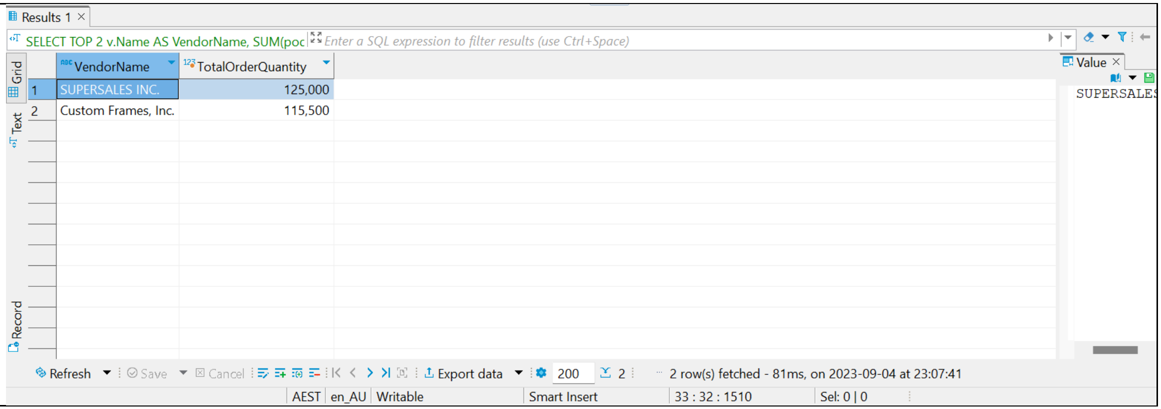
169 row(s) fetched - 96ms (4ms fetch), on 2023-09-10 at 00:57:48

AEST | en_AU | Writable | Smart Insert | 31 : 1 [645] | Sel: 645 | 12

2.2.7. Top Two Vendors by Order Quantity

Code

```
SELECT TOP 2 v.Name AS VendorName, SUM(pod.OrderQty) AS TotalOrderQuantity
FROM Purchasing.PurchaseOrderHeader poh
JOIN Purchasing.PurchaseOrderDetail pod
ON pod.PurchaseOrderID = poh.PurchaseOrderID
JOIN Purchasing.Vendor v
ON v.BusinessEntityID = poh.VendorID
GROUP BY v.Name
ORDER BY sum(pod.OrderQty) DESC
```

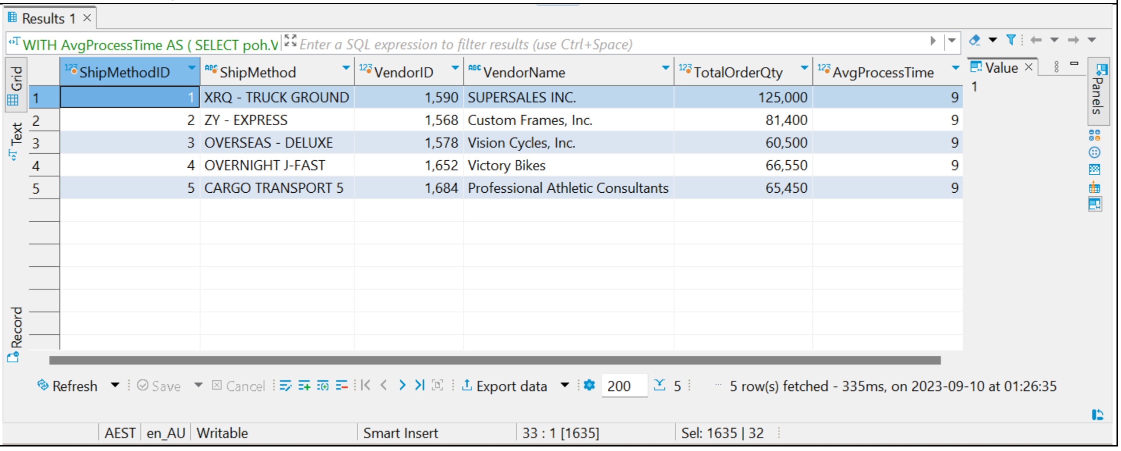
Returned table	 SELECT TOP 2 v.Name AS VendorName, SUM(poc VendorName TotalOrderQuantity 1 SUPERSALES INC. 125,000 2 Custom Frames, Inc. 115,500 2 row(s) fetched - 81ms, on 2023-09-04 at 23:07:41 AEST en_AU Writable Smart Insert 33 : 32 : 1510 Sel: 0 | 0

2.2.8. Identifying the Best Vendors by Shipping Type: Criteria Proposal, Justification, and SQL Solution for Optimized Order Processing

Proposed criterion	Select the top five vendors for the five shipping methods based on highest order quantity processed in the lowest average processing time.
Justification	The vendors are selected by the highest order quantity handled for each shipping type. This guarantees that the vendors with the highest sales volume are prioritized. Minimum average processing time is an important variable in determining order fulfillment efficiency. The goal is to determine the top five suppliers for every delivery method who not only handle a large volume of orders but also finish the deal in the shortest period of time, showing effectiveness in their operations. It will help to increase the company's sales and reputation.
Code	WITH AvgProcessTime AS (SELECT poh.VendorID, AVG(DATEDIFF(DAY, poh.OrderDate, poh.ShipDate)) AS AvgProcessTime, sm.ShipMethodID FROM Purchasing.PurchaseOrderHeader poh JOIN Purchasing.ShipMethod sm ON poh.ShipMethodID = sm.ShipMethodID JOIN Purchasing.PurchaseOrderDetail pod ON poh.PurchaseOrderID = pod.PurchaseOrderID GROUP BY poh.VendorID, sm.ShipMethodID), TotalOrderQuantity AS (SELECT poh.VendorID as VendorId, poh.ShipMethodID as ShipMethodID, sm.Name as ShipMethod, SUM(pod.OrderQty) AS TotalOrderQty FROM Purchasing.PurchaseOrderDetail pod JOIN Purchasing.PurchaseOrderHeader poh ON poh.PurchaseOrderID = pod.PurchaseOrderID JOIN Purchasing.ShipMethod sm ON sm.ShipMethodID = poh.ShipMethodID GROUP BY poh.VendorID, poh.ShipMethodID, sm.Name), RankedVendors AS (SELECT TotalOrderQuantity.ShipMethodID AS ShipMethodID,TotalOrderQuantity.ShipMethod, TotalOrderQuantity.VendorID AS VendorID, TotalOrderQuantity.TotalOrderQty AS TotalOrderQty, AvgProcessTime.AvgProcessTime AS AvgProcessTime, ROW_NUMBER() OVER (PARTITION BY TotalOrderQuantity.ShipMethodID ORDER BY TotalOrderQuantity.TotalOrderQty DESC) AS Rank FROM TotalOrderQuantity JOIN AvgProcessTime ON TotalOrderQuantity.VendorID = AvgProcessTime.VendorID AND TotalOrderQuantity.ShipMethodID = AvgProcessTime.ShipMethodID) SELECT RankedVendors.ShipMethodID, RankedVendors.ShipMethod, RankedVendors.VendorID, v.Name AS VendorName,

```sql
RankedVendors.TotalOrderQty, RankedVendors.AvgProcessTime
FROM RankedVendors
JOIN Purchasing.Vendor v
ON RankedVendors.VendorID = v.BusinessEntityID
WHERE Rank = 1
ORDER BY ShipMethodID ASC;
```

Returned table



Results 1
WITH AvgProcessTime AS (SELECT poh.V
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
Record
Value
1
Panels
Refresh
Save
Cancel
Export data
200
5

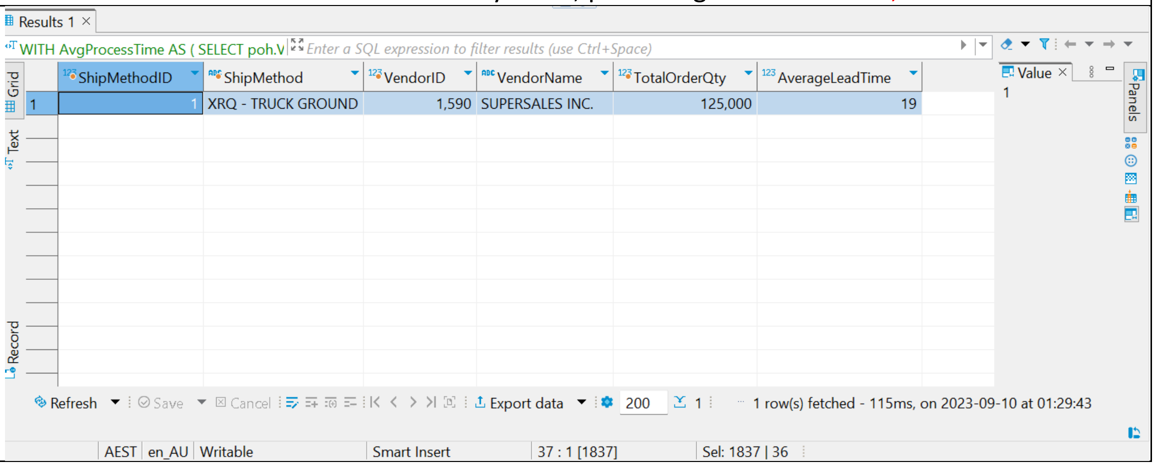
	ShipMethodID	ShipMethod	VendorID	VendorName	TotalOrderQty	AvgProcessTime
1	1	XRQ - TRUCK GROUND	1,590	SUPERSALES INC.	125,000	9
2	2	ZY - EXPRESS	1,568	Custom Frames, Inc.	81,400	9
3	3	OVERSEAS - DELUXE	1,578	Vision Cycles, Inc.	60,500	9
4	4	OVERNIGHT J-FAST	1,652	Victory Bikes	66,550	9
5	5	CARGO TRANSPORT 5	1,684	Professional Athletic Consultants	65,450	9

```
5 row(s) fetched - 335ms, on 2023-09-10 at 01:26:35
AEST | en_AU | Writable | Smart Insert | 33 : 1 [1635] | Sel: 1635 | 32
```

2.2.9. Display the vendor that have sold the highest orders and lowest lead time that is the best vendor.

Code

```sql
WITH AvgProcessTime AS (
SELECT poh.VendorID,
AVG(DATEDIFF(DAY, poh.OrderDate, poh.ShipDate)) AS AvgProcessTime, sm.ShipMethodID
FROM Purchasing.PurchaseOrderHeader poh
JOIN Purchasing.ShipMethod sm
ON poh.ShipMethodID = sm.ShipMethodID
JOIN Purchasing.PurchaseOrderDetail pod ON poh.PurchaseOrderID = pod.PurchaseOrderID
GROUP BY poh.VendorID, sm.ShipMethodID
),
TotalOrderQuantity AS (
SELECT poh.VendorID as VendorId, poh.ShipMethodID as ShipMethodID, sm.Name as
ShipMethod, SUM(pod.OrderQty) AS TotalOrderQty
FROM   Purchasing.PurchaseOrderDetail pod
JOIN Purchasing.PurchaseOrderHeader poh
ON poh.PurchaseOrderID = pod.PurchaseOrderID
JOIN Purchasing.ShipMethod sm
ON sm.ShipMethodID = poh.ShipMethodID
GROUP BY poh.VendorID, poh.ShipMethodID, sm.Name),
RankedVendors AS (
SELECT TotalOrderQuantity.ShipMethodID AS ShipMethodID,TotalOrderQuantity.ShipMethod,
TotalOrderQuantity.VendorID AS VendorID,
TotalOrderQuantity.TotalOrderQty AS TotalOrderQty, AvgProcessTime.AvgProcessTime AS
AvgProcessTime,
ROW_NUMBER() OVER (PARTITION BY TotalOrderQuantity.ShipMethodID ORDER BY
TotalOrderQuantity.TotalOrderQty DESC) AS Rank
FROM TotalOrderQuantity
JOIN AvgProcessTime
ON TotalOrderQuantity.VendorID = AvgProcessTime.VendorID
AND TotalOrderQuantity.ShipMethodID = AvgProcessTime.ShipMethodID)
SELECT TOP 1 RankedVendors.ShipMethodID as ShipMethodID,RankedVendors.ShipMethod,
RankedVendors.VendorID,
v.Name AS VendorName, RankedVendors.TotalOrderQty, pv.AverageLeadTime
FROM RankedVendors
JOIN Purchasing.Vendor v
ON RankedVendors.VendorID = v.BusinessEntityID
```

<table>
<tr><td></td><td>JOIN Purchasing.ProductVendor pv
ON RankedVendors.VendorID = pv.BusinessEntityID
JOIN Purchasing.ShipMethod sm
ON sm.ShipMethodID = RankedVendors.ShipMethodID
ORDER BY RankedVendors.TotalOrderQty DESC, pv.AverageLeadTime ASC;</td></tr>
<tr><td>Returned table</td><td>

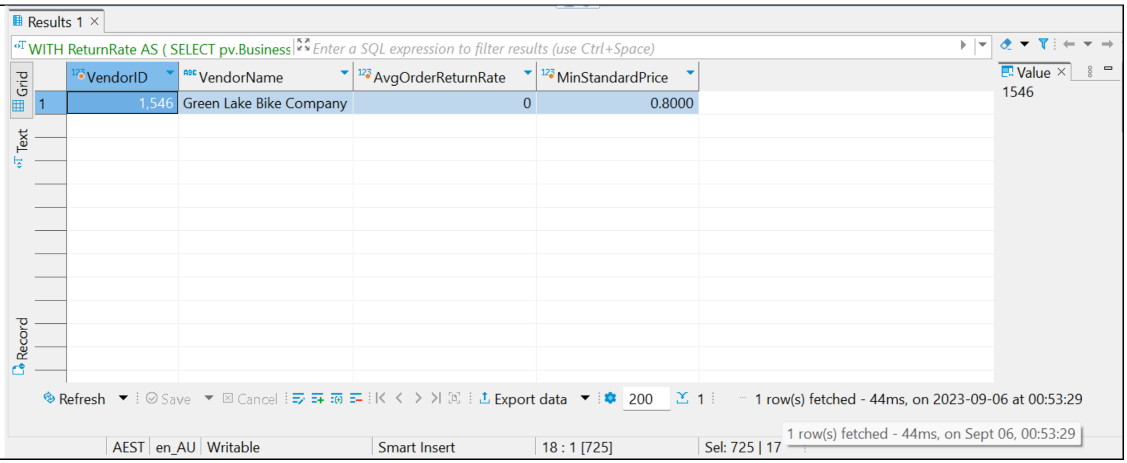
WITH AvgProcessTime AS (SELECT poh.V
Enter a SQL expression to filter results (use Ctrl+Space)
<table>
<tr><th></th><th>ShipMethodID</th><th>ShipMethod</th><th>VendorID</th><th>VendorName</th><th>TotalOrderQty</th><th>AverageLeadTime</th></tr>
<tr><td>1</td><td>1</td><td>XRQ - TRUCK GROUND</td><td>1,590</td><td>SUPERSALES INC.</td><td>125,000</td><td>19</td></tr>
</table>
200 1 1 row(s) fetched - 115ms, on 2023-09-10 at 01:29:43
AEST en_AU Writable Smart Insert 37 : 1 [1837] Sel: 1837 | 36</td></tr>
</table>

2.2.10. Finding the Most Efficient and Cost-Effective Vendor: Criteria Definition and SQL Solution for Optimal Selection

Proposed criterion	Select the best vendor that has the smallest order return rate and the cheapest standard price.
Justification	Low return rates tend to be indicative of great product quality, customer happiness, and efficient customer service. The Lower standard cost might make a vendor more appealing to clients and contribute to their market competitiveness. The goal is to choose a vendor who not only has a solid track record of reducing order returns but also offers affordable pricing. When it comes to quality and pricing, this vendor is likely to be the greatest choice for the organization.

Code	WITH ReturnRate AS (
	SELECT pv.BusinessEntityID AS VendorID, AVG(pod.RejectedQty / pod.OrderQty) AS AvgOrderReturnRate
	FROM Purchasing.ProductVendor pv
	JOIN Purchasing.PurchaseOrderDetail pod
	ON pv.ProductID = pod.ProductID
	GROUP BY pv.BusinessEntityID),
	VendorCheapestPrice AS (
	SELECT pv.BusinessEntityID AS VendorID, MIN(pv.StandardPrice) AS MinStandardPrice
	FROM Purchasing.ProductVendor pv
	GROUP BY pv.BusinessEntityID)
	SELECT TOP 1 a.VendorID, v.Name AS VendorName, a.AvgOrderReturnRate, b.MinStandardPrice
	FROM ReturnRate a
	JOIN VendorCheapestPrice b
	ON a.VendorID = b.VendorID
	JOIN Purchasing.Vendor v
	ON b.VendorID = v.BusinessEntityID
	ORDER BY a.AvgOrderReturnRate ASC, b.MinStandardPrice ASC;

Returned table



Results 1
WITH ReturnRate AS (SELECT pv.Business
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
Record
Value
1546
Refresh
Save
Cancel
Export data
200
1
AEST
en_AU
Writable
Smart Insert
18 : 1 [725]
Sel: 725 | 17
1 row(s) fetched - 44ms, on Sept 06, 00:53:29

	VendorID	VendorName	AvgOrderReturnRate	MinStandardPrice
1	1,546	Green Lake Bike Company	0	0.8000

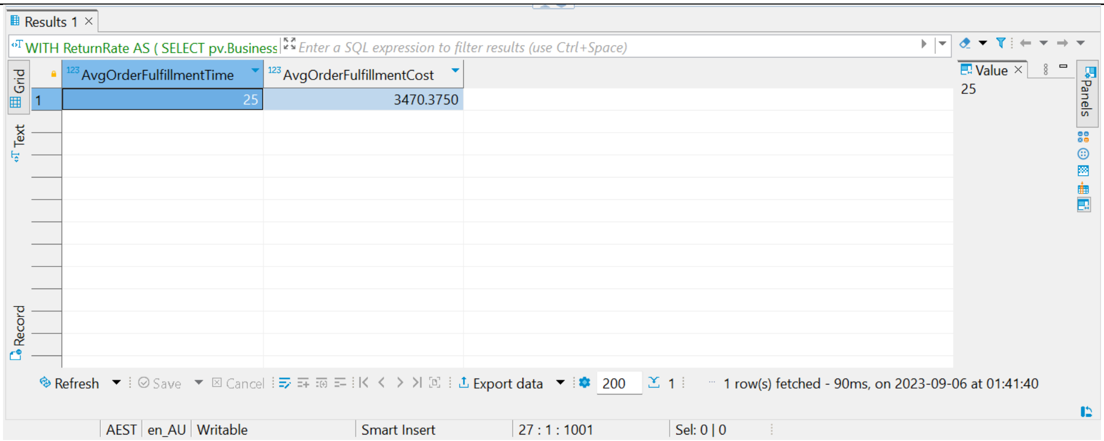
1 row(s) fetched - 44ms, on 2023-09-06 at 00:53:29

2.2.11. Calculate the average order fulfillment cost and time

Code

```sql
WITH ReturnRate AS (
SELECT pv.BusinessEntityID AS VendorID, AVG(pod.RejectedQty / pod.OrderQty) AS
AvgOrderReturnRate
FROM Purchasing.ProductVendor pv
JOIN Purchasing.PurchaseOrderDetail pod
ON pv.ProductID = pod.ProductID
GROUP BY pv.BusinessEntityID),
VendorCheapestPrice AS (
SELECT pv.BusinessEntityID AS VendorID,
    MIN(pv.StandardPrice) AS MinStandardPrice
FROM Purchasing.ProductVendor pv
GROUP BY pv.BusinessEntityID),
MostEffectiveVendor AS (
SELECT TOP 1 a.VendorID
FROM ReturnRate a
JOIN VendorCheapestPrice b
ON a.VendorID = b.VendorID
ORDER BY a.AvgOrderReturnRate ASC, b.MinStandardPrice ASC),
OrderFulfillment AS (
SELECT AVG(po.Freight + po.TaxAmt) AS AvgOrderFulfillmentCost,
AVG(DATEDIFF(day, po.OrderDate, po.ShipDate)) AS AvgOrderFulfillmentTime
FROM Purchasing.PurchaseOrderHeader po
WHERE po.VendorID = (SELECT VendorID FROM MostEffectiveVendor))
SELECT OrderFulfillment.AvgOrderFulfillmentTime, AvgOrderFulfillmentCost
FROM OrderFulfillment;
```

Returned table



	AvgOrderFulfillmentTime	AvgOrderFulfillmentCost
1	25	3470.3750

1 row(s) fetched - 90ms, on 2023-09-06 at 01:41:40