

Certification Project II

Microsoft SQL Server

Internship Program: Data Science and Machine Learning Program

Student: Akram M'Tir

Date: March 2025

Table of Contents

Database creation.....	2
Tables creation	3
Check Database and table creation	5
Data insertion	6
Check Primary Foreign Keys and Indexes	7
ER Diagram.....	9
6 Find Out the Highest and Lowest Amount	10
7 Get the Unique Count of customerName from customers	11
8 Create a View cust_payment	12
9 Create a Stored Procedure for Classic Cars Product Line.....	15
10 Create a Function to Get creditLimit of Customers Less Than 96800.....	16
Difference Between Stored Procedure and Function	16
11 Create a Trigger to Store Transaction Record for Employee Table.....	17
12 Create a Trigger to Display Customer Number If Amount > 10,000.....	19
13 Create Users, Roles, and Logins.....	20
14 Schedule a Job for Database Backup	22
Msdb in SQL Server?.....	22
Enable SQL Server Agent	23
Manually Run the Job to test.....	24
Verify If the Backup Was Created	24
15. Open Activity Monitor & List Observations.....	28
16. Migrate SQL Server to Azure.....	30

Database creation

New Database

Select a page

General

Options

Filegroups

Script

Help

Database name:

MotorsCertification

Owner:

<default>

...

☒ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize	Path
MotorsCertifi...	ROWS ...	PRIMARY	8	By 64 MB, Unlimited	...
MotorsCertifi...	LOG	Not Applicable	8	By 64 MB, Unlimited	...

Server:

LAPTOP-85MTF1DV

Connection:

LAPTOP-85MTF1DV\akram

View connection properties

Progress

Ready

Add

Remove

OK

Cancel

About SQL Server Management Studio

SQL Server Management Studio

20.2

Component Name	Versions
SQL Server Management Studio	20.2.30.0
SQL Server Management Objects (SMO)	17.100.40.0+f571...
Microsoft T-SQL Parser	17.2.3.1+4611522...
Microsoft Analysis Services Client Tools	20.0.3.0
Microsoft Data SqlClient (MDS)	5.1.5
Microsoft SQL Server Data-Tier Application Framework (DacFX)	162.3.566.1+89d8...
Microsoft .NET Framework	4.0.30319.42000
Operating System	10.0.22631

To copy component name and version information, click Copy Info.

Copy Info

Warning: This computer program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law.

© 2024 Microsoft Corporation. All rights reserved.

OK

Tables creation

The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the connection to 'LAPTOP-85MTF1DV.MotorsCertification (LAPTOP-85MTF1DV\akram (57))'. The 'Object Explorer' on the left shows the 'MotorsCertification' database selected, with its 'Tables' folder expanded, listing system tables, user tables (like customers, employees, offices, orders), and dropped ledger tables. The main query window on the right contains the following SQL script:

```
-- SQL Script Part 1

-- 1. Create the Database
CREATE DATABASE MotorsCertification;
USE MotorsCertification;

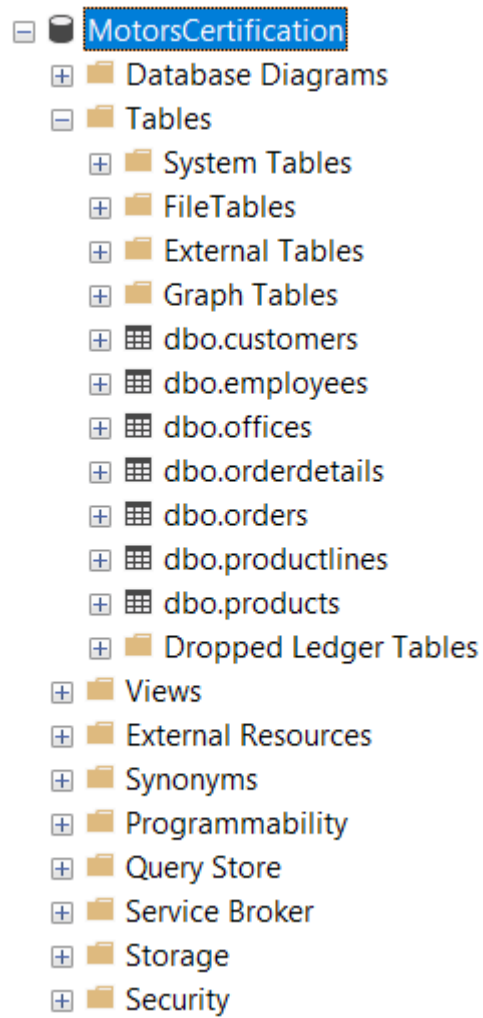
-- 2. Create Tables
-- 2.1. Create offices table first since employees reference it
CREATE TABLE offices (
    officeCode VARCHAR(50) PRIMARY KEY,
    city VARCHAR(50),
    phone VARCHAR(50),
    addressLine1 VARCHAR(50),
    addressLine2 VARCHAR(50),
    state VARCHAR(50),
    country VARCHAR(50),
    postalCode VARCHAR(50),
    territory VARCHAR(50)
);

-- 2.2. Create employees table
CREATE TABLE employees (
    employeeNumber INT PRIMARY KEY,
    lastName VARCHAR(50),
    firstName VARCHAR(50),
    extension VARCHAR(50),
    email VARCHAR(100),
    officeCode VARCHAR(50),
    reportsTo INT,
    jobTitle VARCHAR(50),
    FOREIGN KEY (reportsTo) REFERENCES employees(employeeNumber),
    FOREIGN KEY (officeCode) REFERENCES offices(officeCode)
);

-- 2.3. Create customers table
CREATE TABLE customers (
```

The status bar at the bottom shows 'Item(s) Saved', 'Ln 7', and 'Col 1'. A yellow banner at the bottom of the query window indicates 'Connected. (1/1)'.

Check Database and table creation



Data insertion

The screenshot displays the Microsoft SQL Server Enterprise Studio interface. The top menu bar includes File, Edit, View, Project, Tools, Window, and Help. The toolbar contains various icons for file operations, execution, and navigation. The main window is titled 'SQL_Script_Part2_2 - LAPTOP-85MTF1DV.MotorsCertification (74)' and shows a T-SQL script with two main sections: inserting data into the 'offices' table and the 'employees' table. The 'offices' table insert includes columns for officeCode, city, phone, addressline1, addressline2, state, country, postalCode, and territory. The 'employees' table insert includes columns for employeeNumber, lastName, firstName, extension, email, officeCode, reportsTo, and jobTitle. The script is executed, and the 'Results' pane shows a list of orders with columns for order number, date, status, and customer details. The 'Messages' pane shows a successful query execution message.

SQL_Script_Part2_2 - LAPTOP-85MTF1DV.MotorsCertification (74)

```

-- T-SQL Part2
-- 1. Insert Data into offices Table
INSERT INTO offices (officeCode, city, phone, addressline1, addressline2, state, country, postalCode, territory)
VALUES
('001', 'New York', '123-456-7890', '123 Main St', NULL, 'NY', 'USA', '10001', 'Northeast'),
('002', 'Los Angeles', '987-654-3210', '456 Sunset Blvd', NULL, 'CA', 'USA', '90001', 'West'),
('003', 'Chicago', '312-555-0198', '789 Lake Shore Dr', NULL, 'IL', 'USA', '60601', 'Midwest'),
('004', 'Houston', '713-555-0187', '101 Main St', NULL, 'TX', 'USA', '77001', 'South'),
('005', 'Miami', '305-555-0176', '202 Ocean Dr', NULL, 'FL', 'USA', '33101', 'Southeast'),
('006', 'Seattle', '206-555-0165', '303 Rainy St', NULL, 'WA', 'USA', '98101', 'Northwest'),
('007', 'San Francisco', '415-555-0154', '404 Golden Gate', NULL, 'CA', 'USA', '94101', 'West'),
('008', 'Boston', '617-555-0143', '505 Harvard Ave', NULL, 'MA', 'USA', '02101', 'Northeast'),
('009', 'Dallas', '214-555-0132', '606 Cowboy Rd', NULL, 'TX', 'USA', '75201', 'South'),
('010', 'Denver', '303-555-0121', '707 Rocky Rd', NULL, 'CO', 'USA', '80201', 'Midwest');

-- 2. Insert Data into employees Table
INSERT INTO employees (employeeNumber, lastName, firstName, extension, email, officeCode, reportsTo, jobTitle)
VALUES
(10001, 'Garcia', 'Jorge', '54321', 'jorge.garcia@motorscert.com', '001', NULL, 'Mechanic')

```

Results






























Order Number	Date	Status	Customer Name	Year		
4	2025-03-04	2025-03-08	2025-03-07	Shipped	Customer satisf...	2004
5	2025-03-05	2025-03-09	NULL	Pending	Delayed due to ...	2005
6	2025-03-06	2025-03-10	2025-03-09	Shipped	Express delivery	2006
7	2025-03-07	2025-03-11	2025-03-10	Shipped	Delivered before...	2007
8	2025-03-08	2025-03-12	NULL	Pending	Pending stock a...	2008
9	2025-03-09	2025-03-13	2025-03-12	Shipped	Customer reques...	2009
10	2025-03-10	2025-03-14	NULL	Pending	Awaiting dispatch	2010

Messages

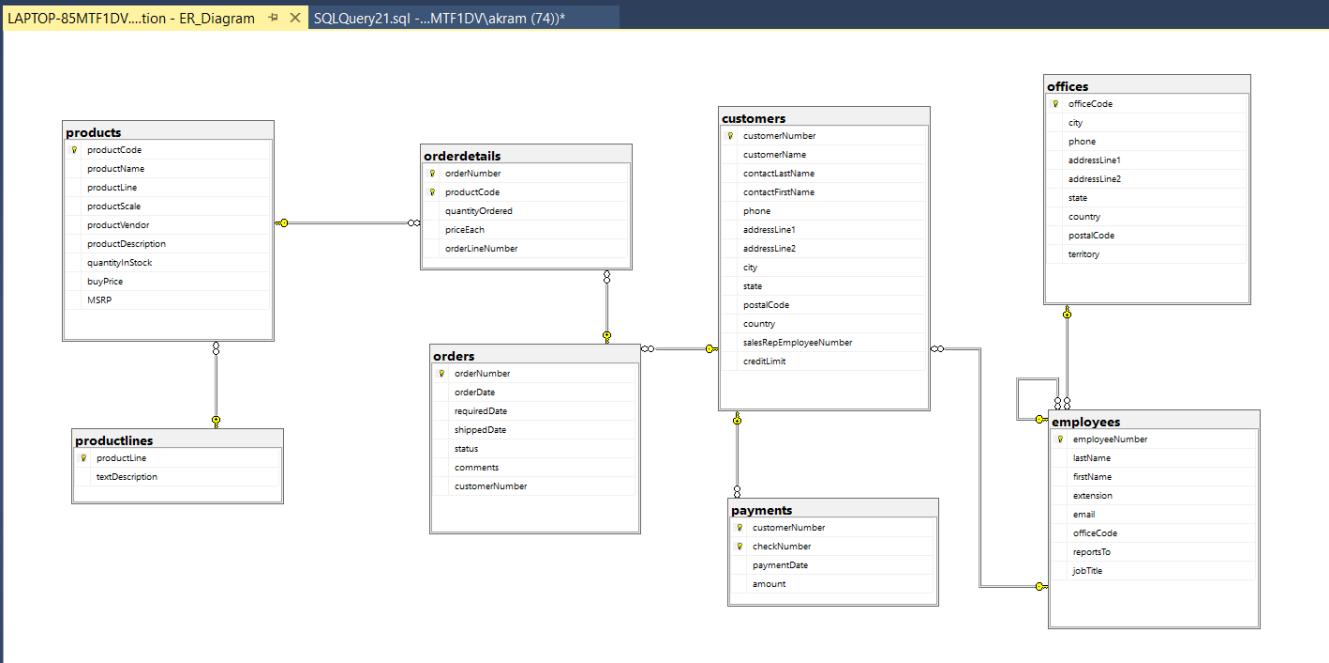
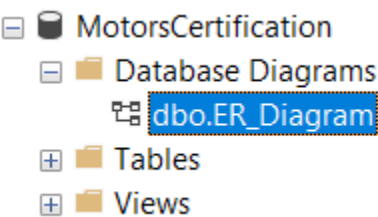
Product Line	Text Description
1	ATVs
2	Bikes
3	Cars
4	Classic Cars
5	Electric Vehicles
6	Hybrid Vehicles
7	Motorcycles
8	Motorbikes
9	Motorcycles
10	Motorcycles

Query executed successfully.

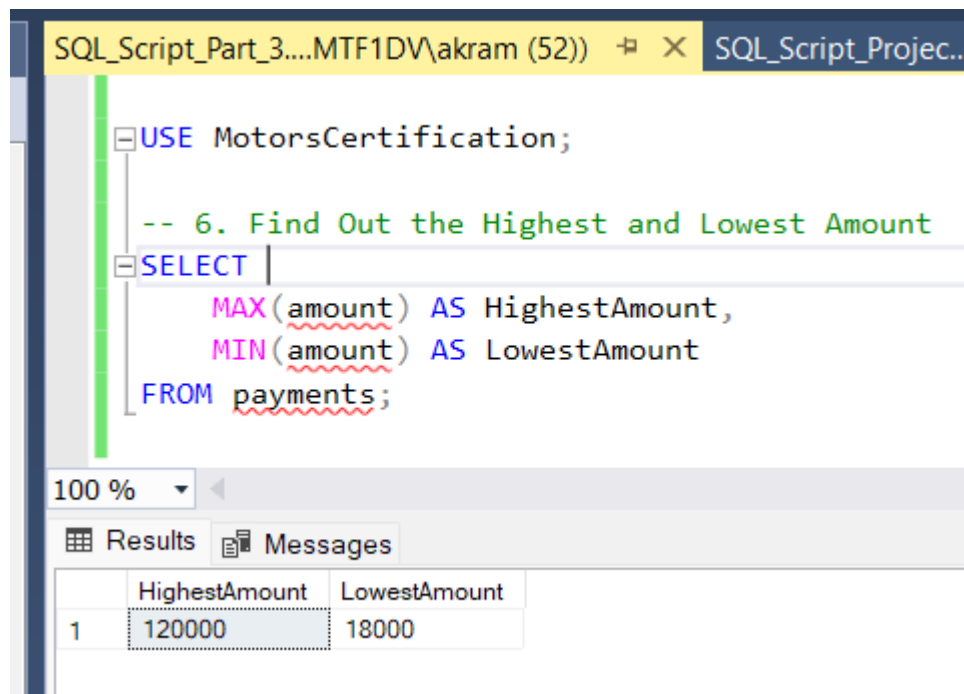
Check Primary Foreign Keys and Indexes

- [-]  dbo.customers
 - [+]  Columns
 - [-]  Keys
 -  PK_customer_6B6369961E8A0D
 -  FK_customers_sales_4316F928
 -  Constraints
 -  Triggers
 - [-]  Indexes
 -  PK_customer_6B6369961E8A0D
 - [+]  Statistics
- [-]  dbo.employees
 - [+]  Columns
 - [-]  Keys
 -  PK_employee_CB72B2356BB1A
 -  FK_employees_offic_403A8C7C
 -  FK_employees_repor_3F466844
 - [+]  Constraints
 - [+]  Triggers
 - [-]  Indexes
 -  PK_employee_CB72B2356BB1A
 - [+]  Statistics
- [+]  dbo.offices
- [+]  dbo.orderdetails
- [+]  dbo.orders
- [-]  dbo.productlines
 - [+]  Columns
 - [-]  Keys
 -  PK_productl_B847E49979146CC
 -  Constraints

ER Diagram



6 Find Out the Highest and Lowest Amount



The screenshot shows a SQL Server Enterprise Manager window with a query script. The script is as follows:

```
USE MotorsCertification;

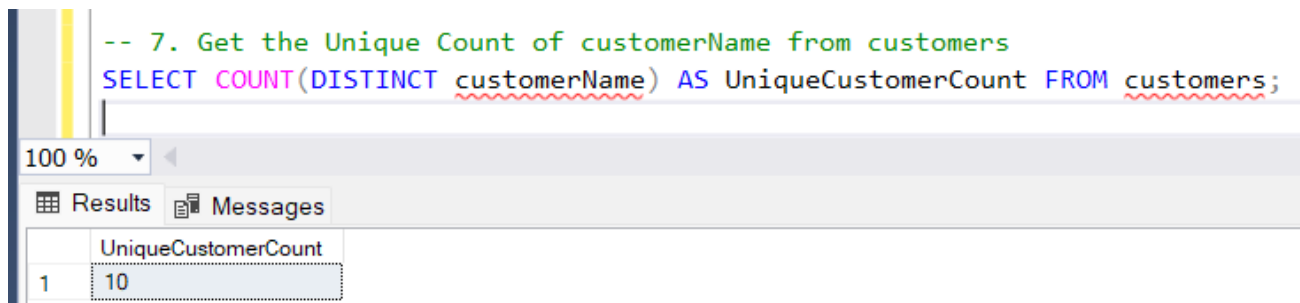
-- 6. Find Out the Highest and Lowest Amount

SELECT
    MAX(amount) AS HighestAmount,
    MIN(amount) AS LowestAmount
FROM payments;
```

Below the script, the 'Results' tab is active, displaying the following data:

	HighestAmount	LowestAmount
1	120000	18000

7 Get the Unique Count of customerName from customers



The screenshot shows a SQL Server Enterprise Manager window with a query script. The script is as follows:

```
-- 7. Get the Unique Count of customerName from customers
SELECT COUNT(DISTINCT customerName) AS UniqueCustomerCount FROM customers;
```

Below the script, the 'Results' tab is active, displaying the following data:

	UniqueCustomerCount
1	10

8 Create a View cust_payment

```
-- 8. Create a View cust_payment
CREATE VIEW cust_payment AS
SELECT
    c.customerName,
    p.amount,
    c.contactLastName,
    c.contactFirstName
FROM payments p
JOIN customers c ON p.customerNumber = c.customerNumber;

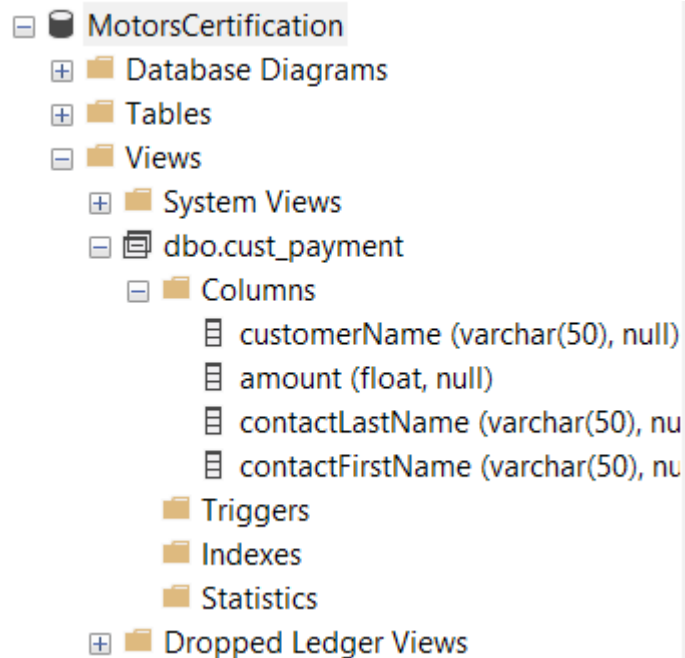
-- Select from the View
SELECT * FROM cust_payment;

-- Truncate and Drop the View
DROP VIEW cust_payment;
```

100 %

Results Messages

	customerName	amount	contactLastName	contactFirstName
1	Hercules Garage	30000	Doe	Jane
2	Speedy Motors	18000	Johnson	Michael
3	Luxury Cars Ltd	75000	Brown	Emily
4	MotoHub	120000	Wilson	Daniel
5	Elite Auto	95000	Davis	Sarah
6	Turbo Cars	84000	Martinez	Laura
7	Superbike World	35000	Anderson	James
8	Hyper Cars	34000	Thomas	Patricia
9	Moto Enthusiasts	28000	Hernandez	Robert
10	HighSpeed Auto	60000	Gonzalez	David



9 Create a Stored Procedure for Classic Cars Product Line

This stored procedure will display productLine for "Classic Cars".

```
-- 9. Create a Stored Procedure for Classic Cars Product Line
CREATE PROCEDURE GetClassicCars
AS
BEGIN
    SELECT * FROM products WHERE productLine = 'Classic Cars';
END;

-- Execute the Stored Procedure
EXEC GetClassicCars;
```

100 %

Results Messages

	productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP
1	P007	Chevrolet Camaro	Classic Cars	1:18	Chevrolet	American muscle car.	20	35000	38000

10 Create a Function to Get creditLimit of Customers Less Than 96800

This function will return a table of customers with creditLimit less than 96,800.

```
-- 9. Create a Stored Procedure for Classic Cars Product Line
CREATE PROCEDURE GetClassicCars
AS
BEGIN
    SELECT * FROM products WHERE productLine = 'Classic Cars';
END;

-- Execute the Stored Procedure
EXEC GetClassicCars;
```

100 %

Results Messages

	productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP
1	P007	Chevrolet Camaro	Classic Cars	1:18	Chevrolet	American muscle car.	20	35000	38000

Difference Between Stored Procedure and Function

Feature	Stored Procedure	Function
Purpose	Performs an action (e.g., inserting, updating, or deleting data)	Returns a value or table (used for calculations or queries)
Return Type	Can return multiple result sets or nothing	Must return a single value or a table
Usage in Queries	Cannot be used inside SELECT statements	Can be used inside SELECT, WHERE, JOIN
Can Modify Data?	✓Yes (INSERT, UPDATE, DELETE)	✗No (Only SELECT is allowed)
Calling Method	EXEC ProcedureName	SELECT * FROM FunctionName()
Example Use Case	Retrieve product details, process orders, update stock	Calculate tax, filter customers by credit limit

11 Create a Trigger to Store Transaction Record for Employee Table

This trigger will store a transaction record whenever a **new employee** is inserted. It will log employeeNumber, lastName, firstName, and officeCode.

```
SQL_Script_Part_3....MTF1DV\akram (52)) * X SQL_Script_Projec...MTF1DV\akram (57)) LAPTOP-85MTF1DV....tion - ER_Diagram SQL_Script

-- 11. Create a Trigger to Store Transaction Record for Employee Table
-- Step 1: Create an Audit Table
CREATE TABLE EmployeeAudit (
    auditID INT IDENTITY(1,1) PRIMARY KEY,
    employeeNumber INT,
    lastName VARCHAR(50),
    firstName VARCHAR(50),
    officeCode VARCHAR(50),
    insertedAt DATETIME DEFAULT GETDATE()
);

-- Step 2: Create the Trigger
CREATE TRIGGER trg_Employee_Insert
ON employees
AFTER INSERT
AS
BEGIN
    INSERT INTO EmployeeAudit (employeeNumber, lastName, firstName, officeCode)
    SELECT employeeNumber, lastName, firstName, officeCode FROM inserted;

    PRINT 'Employee transaction recorded successfully.';
END;

-- Insert a New Employee (Trigger Will Fire Automatically)
INSERT INTO employees (employeeNumber, lastName, firstName, extension, email, officeCode, reportsTo, jobTitle)
VALUES (1011, 'Williams', 'Sophia', 'x133', 'sophia.williams@example.com', '003', 1001, 'Sales Rep');
-- Check If the Trigger Stored the Transaction in EmployeeAudit
SELECT * FROM EmployeeAudit;
```

100 %

Messages

(1 row affected)
Employee transaction recorded successfully.

(1 row affected)

Completion time: 2025-03-07T23:43:55.8026852+01:00

100 %

Query executed successfully. LAPTOP-85MTF1DV (16.0 RTM) LAPTOP-8


























```
-- Insert a New Employee (Trigger Will Fire Automatically)
INSERT INTO employees (employeeNumber, lastName, firstName, extension, email, officeCode, reportsTo, jobTitle)
VALUES (1011, 'Williams', 'Sophia', 'x133', 'sophia.williams@example.com', '003', 1001, 'Sales Rep');
-- Check If the Trigger Stored the Transaction in EmployeeAudit
SELECT * FROM EmployeeAudit;
```

100 %

Results

Messages

	auditID	employeeNumber	lastName	firstName	officeCode	insertedAt
1	1	1011	Williams	Sophia	003	2025-03-07 23:43:55.797

- [-]  MotorsCertification
 - [+]  Database Diagrams
 - [-]  Tables
 - [+]  System Tables
 - [+]  FileTables
 - [+]  External Tables
 - [+]  Graph Tables
 - [+]  dbo.customers
 - [+]  dbo.EmployeeAudit
 - [-]  dbo.employees
 - [+]  Columns
 - [+]  Keys
 - [+]  Constraints
 - [-]  Triggers
 -  trg_Employee_Insert
 - [+]  Indexes
 - [+]  Statistics
 - [+]  dbo.offices
 - [+]  dbo.orderdetails
 - [+]  dbo.orders
 - [+]  dbo.payments
 - [+]  dbo.productlines
 - [+]  dbo.products
 - [+]  Dropped Ledger Tables
- [+]  Views

12 Create a Trigger to Display Customer Number If Amount > 10,000

Whenever a new **payment** is added, if the amount is greater than **10,000**, the trigger will print the customerNumber.

SQL_Script_Part_3....MTF1DV\akram (52))* X SQL_Script_Projec...MTF1DV\akram (57)) LAPTOP-85MTF1DV....tion -

```
-- Whenever a new payment is added, if the amount is greater than 10,000, the trigger will
CREATE TRIGGER trg_HighPayment
ON payments
AFTER INSERT
AS
BEGIN
    DECLARE @customerNumber INT;

    SELECT @customerNumber = customerNumber FROM inserted WHERE amount > 10000;

    IF @customerNumber IS NOT NULL
    BEGIN
        PRINT 'Customer Number with High Payment: ' + CAST(@customerNumber AS VARCHAR);
    END
END;

-- Insert a Payment Greater Than 10,000 (Trigger Should Fire)
INSERT INTO payments (customerNumber, checkNumber, paymentDate, amount)
VALUES (2005, 'CHK011', '2025-03-12', 9500.00); -- Amount < 10,000
INSERT INTO payments (customerNumber, checkNumber, paymentDate, amount)
VALUES (2006, 'CHK012', '2025-03-13', 15000.00); -- Amount > 10,000

SELECT * FROM sys.triggers WHERE name = 'trg_HighPayment';
```

100 %

Messages

Customer Number with High Payment: 2006

(1 row affected)

Completion time: 2025-03-07T23:53:04.8894180+01:00

Insert a Payment Greater Than 10,000 (Trigger Should Fire)

SELECT * FROM sys.triggers WHERE name = 'trg_HighPayment';

100 %

Results Messages

	name	object_id	parent_class	parent_class_desc	parent_id	type	type_desc	create_date
1	trg_HighPayment	1669580986	1	OBJECT_OR_COLUMN	1333579789	TR	SQL_TRIGGER	2025-03-07 23:51:17.197

13 Create Users, Roles, and Logins

Step 1: Create Logins

Step 2: Create Database Users

Step 3: Create Roles

Step 4: Assign Permissions

The screenshot displays the Microsoft SQL Server Management Studio interface. The title bar indicates the file is 'SQL_Script_Part_3.sql' located on 'LAPTOP-85MTF1DV.MotorsCertification (LAPTOP-85MTF1DV\akram (52))'. The menu bar includes File, Edit, View, Query, Project, Tools, Window, and Help. The toolbar contains various icons for file operations, execution, and navigation. The 'Object Explorer' on the left shows the database structure for 'MotorsCertification', including 'dbo.productlines', 'dbo.products', 'Dropped Ledger Tables', 'Views', 'External Resources', 'Synonyms', 'Programmability', 'Query Store', 'Service Broker', 'Storage', 'Security', 'Users' (with sub-items like AdminUser, dbo, EmployeeUser, guest, HRUser, INFORMATION_SCHEMA, sys), and 'Roles' (with sub-items like Database Roles, AdminRole, db_accessadmin, db_backupoperator, db_datareader, db_datawriter, db_ddladmin, db_denydatareader, db_denydatawriter, db_owner, db_securityadmin, EmployeeRole, HRRole, public, and Application Roles).

The main query window shows the following SQL script:

```
-- Step 1: Create Logins
CREATE LOGIN AdminLogin WITH PASSWORD = 'Admin@123';
CREATE LOGIN HRLogin WITH PASSWORD = 'HR@123';
CREATE LOGIN EmployeeLogin WITH PASSWORD = 'Employee@123';

-- Step 2: Create Database Users
USE MotorsCertification;

CREATE USER AdminUser FOR LOGIN AdminLogin;
CREATE USER HRUser FOR LOGIN HRLogin;
CREATE USER EmployeeUser FOR LOGIN EmployeeLogin;

-- Step 3: Create Roles
CREATE ROLE AdminRole;
CREATE ROLE HRRole;
CREATE ROLE EmployeeRole;

-- Step 4: Assign Permissions
-- Admin Role (Full Access)
GRANT CONTROL ON DATABASE::MotorsCertification TO AdminRole;
ALTER ROLE AdminRole ADD MEMBER AdminUser;

-- HR Role (Access Only employees and offices Tables)
GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO HRRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON offices TO HRRole;
ALTER ROLE HRRole ADD MEMBER HRUser;

-- Employee Role (Read-Only Access to All Tables)
GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO HRRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON offices TO HRRole;
ALTER ROLE HRRole ADD MEMBER HRUser;
```

The 'Messages' pane at the bottom shows the execution results:

```
Commands completed successfully.

Completion time: 2025-03-08T15:44:41.9663078+01:00

Query executed successfully.
```

The status bar at the bottom indicates 'Ready' and shows the current position as 'Ln 147 Col 1'.

- [-] Security
 - [-] Users
 - AdminUser
 - dbo
 - EmployeeUser
 - guest
 - HRUser
 - INFORMATION_SCHEMA
 - sys
 - [-] Roles
 - [-] Database Roles
 - AdminRole
 - db_accessadmin
 - db_backupoperator
 - db_datareader
 - db_datawriter
 - db_ddladmin
 - db_denydatareader
 - db_denydatawriter
 - db_owner
 - db_securityadmin
 - EmployeeRole
 - HRRole
 - public
 - [+] Application Roles
 - [+] Schemas

14 Schedule a Job for Database Backup

This job will **automatically back up the database** on a scheduled basis.

MsdB in SQL Server?

msdb is a **system database** in SQL Server that is used by **SQL Server Agent** for **scheduling jobs, alerts, and backups**.

We are creating a **SQL Server Agent Job** to **automatically back up the database**. SQL Server Agent uses msdb to **store job-related information** such as:

- **Scheduled Jobs**
- **Backup History**
- **Alerts & Notifications**
- **Job Steps and Execution History**

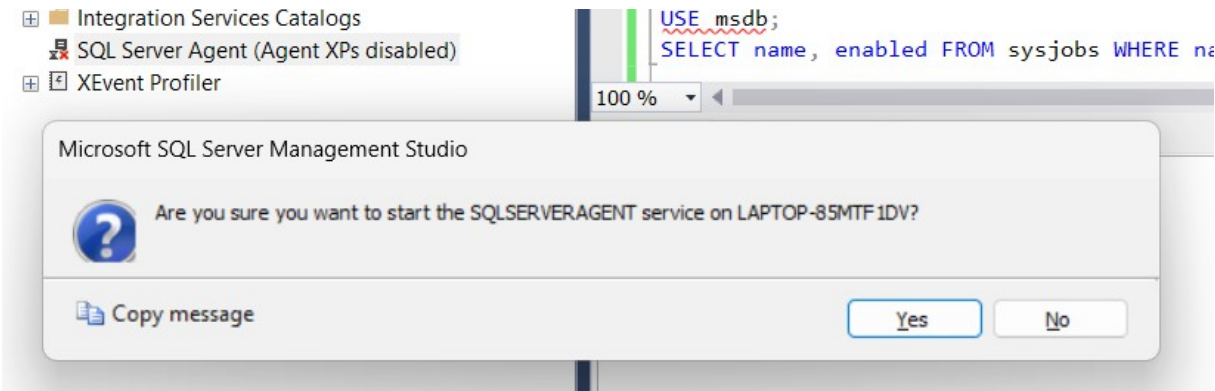
The screenshot displays the Microsoft SQL Server Management Studio interface. The title bar indicates the connection to 'LAPTOP-85MTF1DV.msdB (LAPTOP-85MTF1DV\akram (52))'. The Object Explorer on the left shows the 'SQL Server Agent' folder expanded, with the 'Jobs' folder selected, and the 'MotorsCertification_Backup' job highlighted. The main query window shows the following T-SQL script:

```
USE msdb;
-- Create a Backup Job
EXEC sp_add_job
    @job_name = 'MotorsCertification_Backup';
-- Creates a new scheduled job named MotorsCertification_Backup
-- -- Add a Job Step (Perform Database Backup)
EXEC sp_add_jobstep
    @job_name = 'MotorsCertification_Backup',
    @step_name = 'Full Database Backup',
    @subsystem = 'TSQL',
    @command = 'BACKUP DATABASE MotorsCertification TO DISK = ''C:\SQLBackups\Mot',
    @database_name = 'MotorsCertification';
-- Schedule the Job to Run Daily at 2:00 AM
EXEC sp_add_schedule
    @schedule_name = 'DailyBackupSchedule',
    @freq_type = 4, -- Daily
    @freq_interval = 1, -- Every day
    @active_start_time = 020000; -- Runs at 2:00 AM
-- Attach the Schedule to the Backup Job
EXEC sp_attach_schedule
    @job_name = 'MotorsCertification_Backup',
    @schedule_name = 'DailyBackupSchedule';
-- Assign the Job to the SQL Server Instance
EXEC sp_add_jobserver
    @job_name = 'MotorsCertification_Backup';
-- This will run a daily backup at 2:00 AM.
-- Verify if the Job Exists
USE msdb;
SELECT name, enabled FROM sysjobs WHERE name = 'MotorsCertification_Backup';
```

At the bottom, the 'Results' pane shows the output of the final query:

	name	enabled
1	MotorsCertification_Backup	1

Enable SQL Server Agent

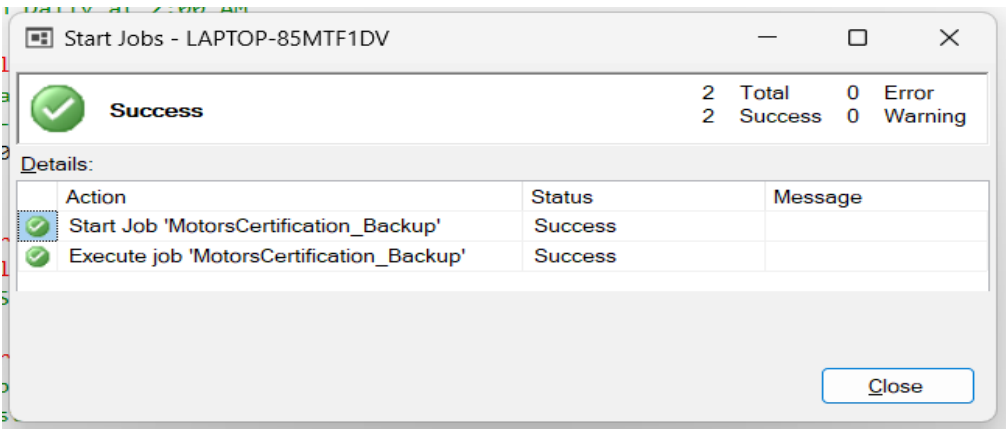


- SQL Server Agent
 - Jobs
 - MotorsCertification_Backup
 - syspolicy_purge_history
 - Job Activity Monitor
 - Alerts
 - Operators
 - Proxies
 - Error Logs
 - XEvent Profiler

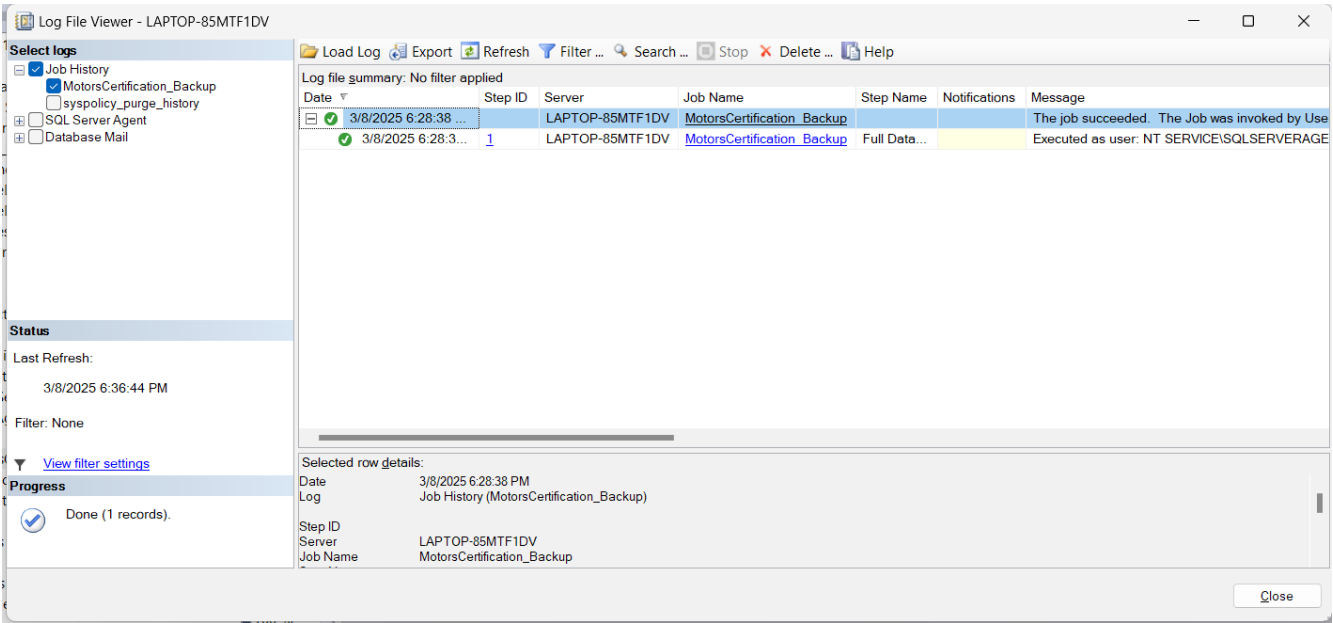
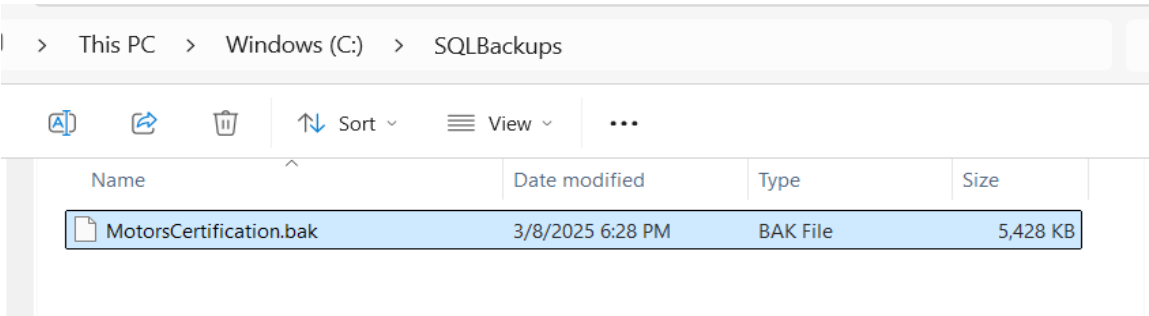
The screenshot shows the SQL Server Configuration Manager window. The left pane shows the 'SQL Server Services' folder expanded. The right pane displays a table with the following data:

Name	State	Start Mode	Log On As	Process ID	Service Type
SQL Server (MSSQLSERVER)	Running	Automatic	NT Service\MSSQLS...	23348	SQL Server
SQL Server Browser	Stopped	Other (Boot, System,...	NT AUTHORITY\LOC...	0	
SQL Server Agent (MSSQLSERVER)	Running	Manual	NT Service\SQLSERV...	54572	SQL Agent

Manually Run the Job to test

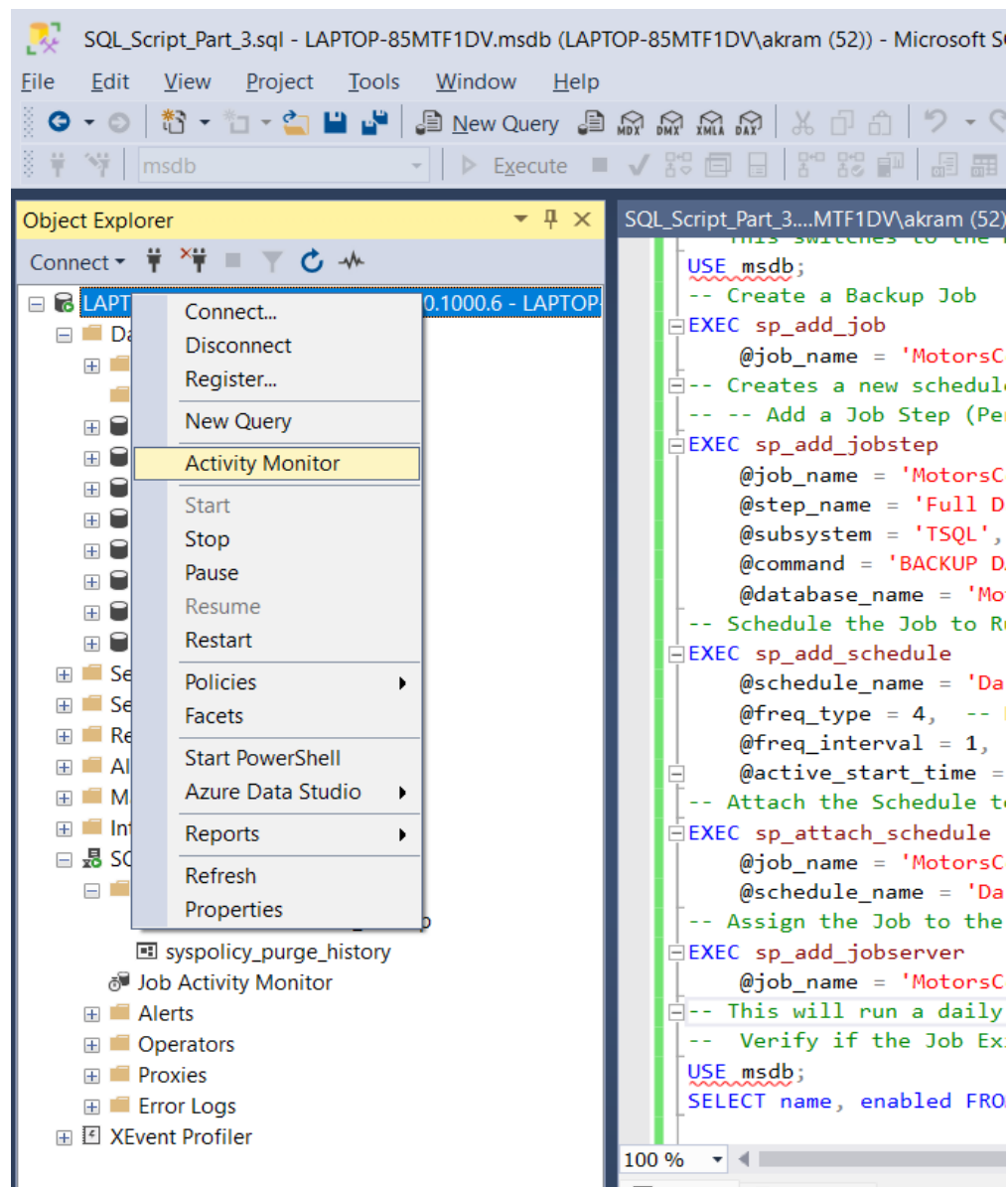


Verify If the Backup Was Created



15. Open Activity Monitor & List Observations

Open Activity Monitor:



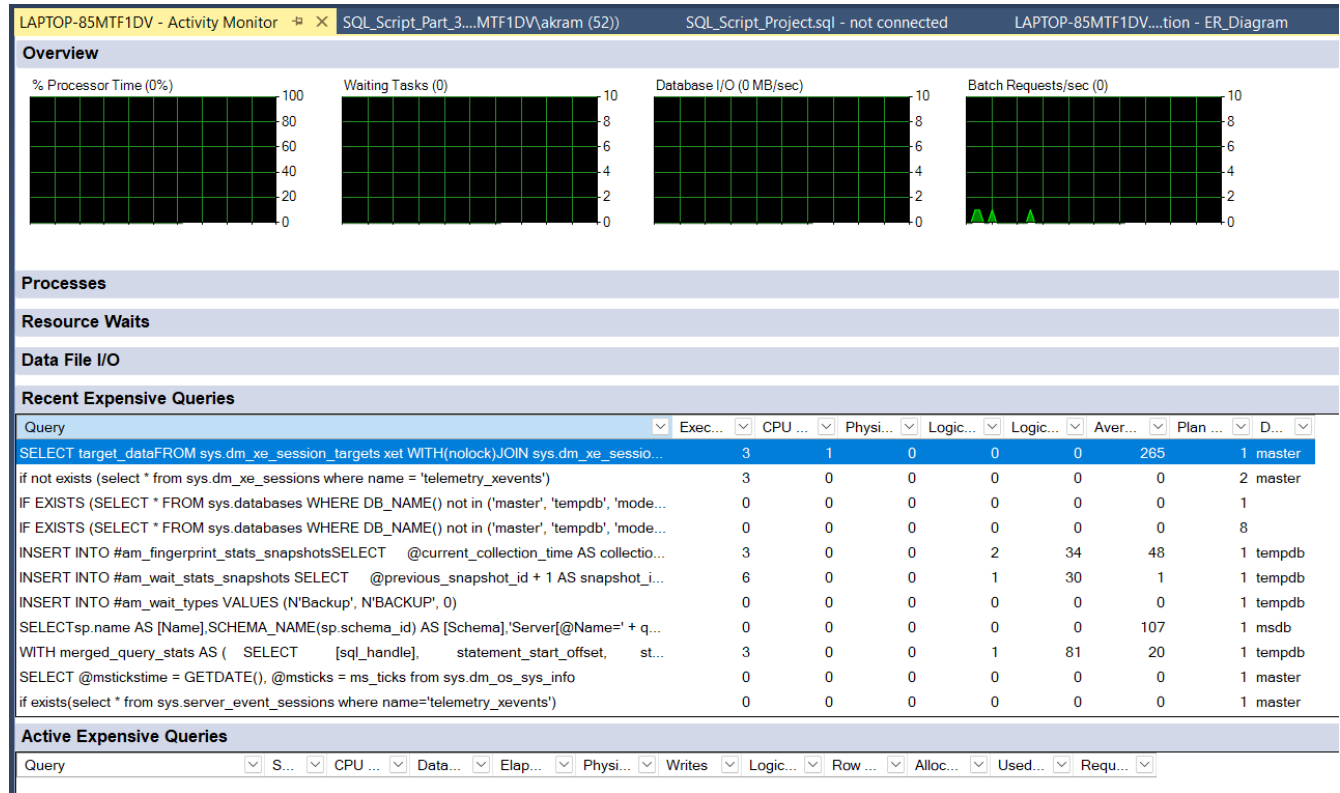
Observations in Activity Monitor:

- **Processes:**
 - Some background system processes are always running.
 - Query execution threads are active based on workload.
- **Resource Waits:**
 - No significant waits in a low-traffic environment.
 - Increased wait times may indicate locks or performance bottlenecks.
- **Active Expensive Queries:**

- Expensive queries usually involve **large joins** or **missing indexes**.
- **CPU-intensive queries** may indicate performance tuning is needed.

Recommendation:

Monitor **CPU usage**, **I/O waits**, and **blocking queries** for optimization.



16. Migrate SQL Server to Azure

Steps to Migrate SQL Server Database to Azure SQL Database

1. Use Azure Data Migration Assistant (DMA)

- Download and install **Azure DMA**.
- Analyze the database for **compatibility issues**.
- Migrate schema and data.

2. Use SQL Server Management Studio (SSMS)

- Right-click the **database** → **Tasks** → **Deploy Database to Microsoft Azure SQL Database**.
- Provide **Azure SQL Server credentials**.
- Complete the migration.

3. Use Azure Database Migration Service (DMS)

- In **Azure Portal**, create a **Migration Service**.
- Select **Online** or **Offline Migration**.
- Provide **source and target database details**.
- Start **data transfer**.

