

Expensify

An Expense And Procurement Management System

(Project for - MSIS 2603 -Database Management Systems)

Team Expensify:

- Garima Jain
- Shalini Gopalakrishnan
- Srivaths Rai
- Soudamini Modak

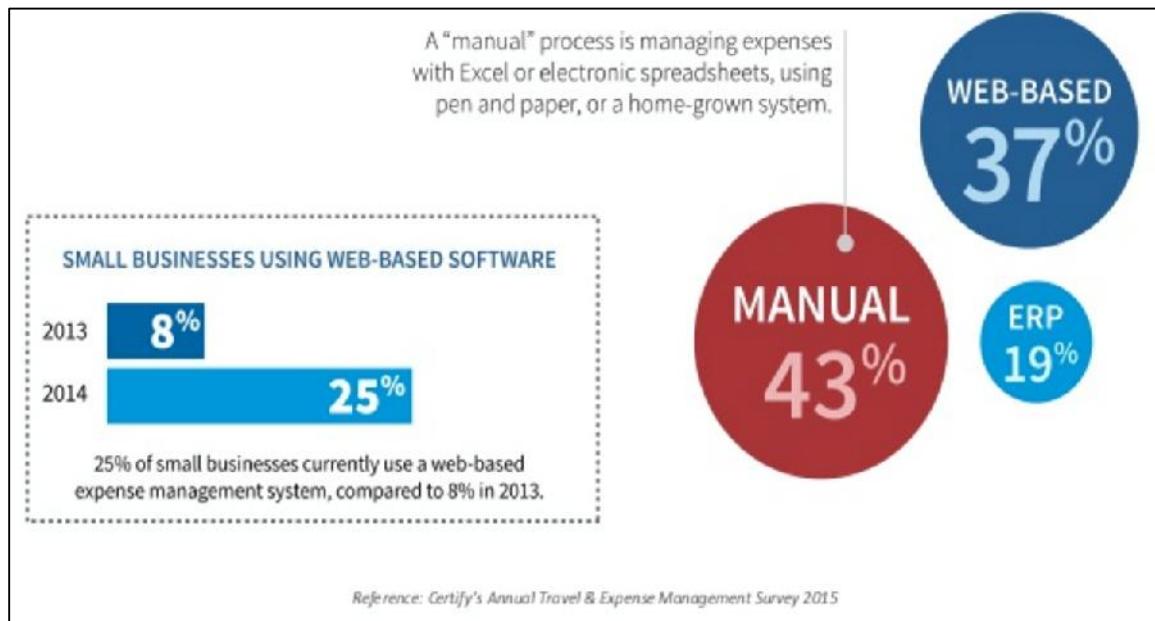
Report Submitted on:
05/31/2016

Table of Contents

| | |
|-------------------------------------------------------------------|----|
| 1. Business Need, Application Description and Business Value..... | 2 |
| 2. Users Of Expensify | 3 |
| 3. Use Cases | 3 |
| 4. UML | 5 |
| 5. Physical Schema – Data Dictionary | 6 |
| 6. SQL Scripts..... | 10 |
| 7. SQL Queries..... | 11 |
| 8. Business Metrcis Queries | 41 |
| 9. Business Metrics Reports..... | 45 |
| 10. Project Summary | 49 |

1. Business Need, Application Description and Business Value

Automation of expense and procurement management is becoming extremely crucial and organizations are rapidly adopting automation for this process. Following data shows trends in the industry.



Expense and Procurement Management system (Expensify) automates the expense claims and purchase requests for employees in the company. Expensify mainly caters to small to medium companies where people go through tedious process of manual application for expense claims (reimbursement request) and purchase request.

Automation in reimbursement and procurement requests brings transparency in the requests as well as approvals. This automation would also bring efficiency to the overall expense and procurement management.

2. Users Of Expensify

- Requestors – Requestor is any employee of the organization using Expensify, who wants to make Purchase request or a Reimbursement request
- Managers – Manager is an employee of the organization, who has authority to approve purchase requests based on the purchase limit of the requester.
- Procurement and Payment Associate – Procurement and Payment Associate is an employee of the organization, who creates purchase orders in the system and makes payment for Invoices.
- Suppliers – Suppliers have restricted access to Expensify wherein they can view purchase orders assigned to them and can update the status as the order progresses.
- Analysts – Analyst is a very important user of Expensify, who would slice and dice the data and draw reports to help decision making.

3. Use Cases

Requestor

- Create a purchase request for items like laptop, keyboard, mouse, projector, dock station etc.
- Cancel the request
- Create a reimbursement request for reimbursement related to business travel, food, accommodation, team outings etc.
- View purchase / reimbursement requests submitted in the past

Manager

- View pending and approved Purchase requests
- Search for a request by a Request ID or by approver's name
- Approve or reject requests made by the requestors based on the requestors' expense limit

Procurement and Payment Associate

- View and verify the approved purchase requests
- Create purchase orders associated with a purchase request
- Track open purchase orders
- Make payments for invoices

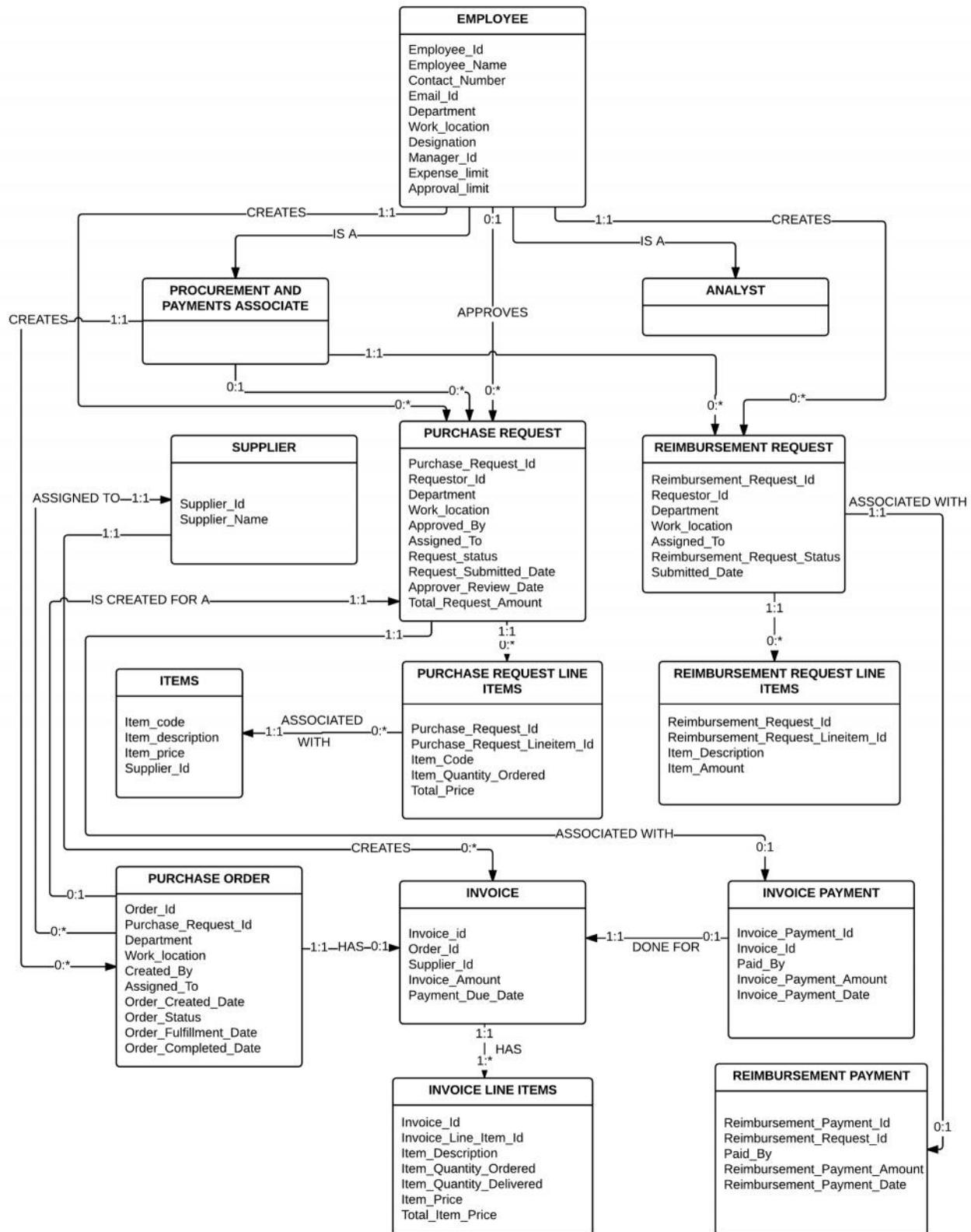
Suppliers

- View and update the order status and order fulfillment date
- View past and pending orders

Analyst

- Draw business metrics reports like – Suppliers by spend, total spend by departments and work locations, overdue payments, delayed deliveries etc.

4. UML Model



5. Physical Schema – Data Dictionary

| EMPLOYEE | | |
|----------------|---------------|----------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK,FK,NULLABLE) |
| EMPLOYEE_ID | INTEGER | PRIMARY KEY, NOT NULL |
| EMPLOYEE_NAME | VARCHAR(70) | NOT NULL |
| EMAIL_ID | VARCHAR(50) | NOT NULL |
| CONTACT_NO | VARCHAR(15) | NULL |
| DESIGNATION | VARCHAR(20) | NOT NULL |
| DEPARTMENT | VARCHAR(20) | NOT NULL |
| WORK_LOCATION | VARCHAR(20) | NOT NULL |
| EXPENSE_LIMIT | DECIMAL(10,2) | NOT NULL |
| APPROVAL_LIMIT | DECIMAL(10,2) | NULL |

| ROLE | | |
|-------------|-------------|----------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK,FK,NULLABLE) |
| ROLE_ID | INTEGER | PRIMARY KEY, NOT NULL |
| ROLE_NAME | VARCHAR(20) | NOT NULL |

| USER_ROLE | | |
|------------------|-----------|-------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK,FK,NULLABLE) |
| USER_ROLE_KEY_ID | INTEGER | PRIMARY KEY, NOT NULL |
| EMPLOYEE_ID | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NOT NULL |
| ROLE_ID | INTEGER | FOREIGN KEY (ROLE -> ROLE_ID), NOT NULL |

| MANAGES | | |
|----------------------------------------|-----------|-------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT (PK, FK,NULLABLE) |
| EMPLOYEE_ID | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NOT NULL |
| MANAGER_ID | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NOT NULL |
| PRIMARY KEY - EMPLOYEE_ID, APPROVER_ID | | |

| SUPPLIER | | |
|-------------------|--------------|------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK, FK, NULLABLE) |
| SUPPLIER_ID | INTEGER | PRIMARY KEY, NOT NULL |
| SUPPLIER_NAME | VARCHAR(70) | NOT NULL |
| SUPPLIER_EMAIL_ID | VARCHAR (50) | NOT NULL |

| PURCHASE REQUEST | | |
|------------------------|----------------|--------------------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT (PK, FK, NULLABLE) |
| PURCHASE REQUEST_ID | INTEGER | PRIMARY KEY, NOT NULL |
| REQUESTOR_ID | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NOT NULL |
| APPROVED_BY | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NULL |
| REQUEST_STATUS | VARCHAR (20) | NOT NULL, DOMAIN VALUES - (OPEN, APPROVED, REJECTED, CLOSED) |
| ASSIGNED_TO | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NULL |
| REQUEST_SUBMITTED_DATE | DATETIME | NOT NULL |
| TOTAL_REQUEST_AMOUNT | DECIMAL (10,2) | NOT NULL |
| APPROVER REVIEW_DATE | DATETIME | NULL |
| DEPARTMENT | VARCHAR (20) | NOT NULL |
| WORK_LOCATION | VARCHAR (20) | NOT NULL |

| REIMBURSEMENT_REQUEST | | |
|------------------------------|--------------|--------------------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT (PK, FK, NULLABLE) |
| REIMBURSEMENT_REQUEST_ID | INTEGER | PRIMARY KEY, NOT NULL |
| REQUESTOR_ID | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NOT NULL |
| ASSIGNED_TO | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NOT NULL |
| SUBMITTED_DATE | DATETIME | NOT NULL |
| REIMBURSEMENT_REQUEST_STATUS | VARCHAR (20) | NOT NULL, DOMAIN VALUES - (OPEN, APPROVED, REJECTED, CLOSED) |
| DEPARTMENT | VARCHAR (20) | NOT NULL |
| WORK_LOCATION | VARCHAR (20) | NOT NULL |

| ITEMS | | |
|------------------|--------------|-----------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK,FK,NULLABLE) |
| ITEM_CODE | VARCHAR (20) | PRIMARY KEY, NOT NULL |
| ITEM_DESCRIPTION | VARCHAR(50) | NOT NULL |
| ITEM_PRICE | DECIMAL(7,2) | NOT NULL |
| SUPPLIER_ID | INTEGER | FOREIGN KEY (SUPPLIER-> SUPPLIER_ID),NOT NULL |

| PURCHASE_REQUEST_LINE_ITEMS | | |
|---------------------------------------------------------------|----------------|-----------------------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT (PK,FK, NULLABLE) |
| PURCHASE_REQUEST_ID | INTEGER | FOREIGN KEY (PURCHASE REQUEST -> PURCHASE_REQUEST_ID), NOT NULL |
| PURCHASE_REQUEST_LINE_ITEM_ID | INTEGER | NOT NULL |
| ITEM_CODE | VARCHAR (20) | FOREIGN KEY (ITEMS -> ITEM_CODE), NOT NULL |
| ITEM_QUANTITY_ORDERED | INTEGER | NOT NULL |
| TOTAL_PRICE | DECIMAL (10,2) | NOT NULL |
| PRIMARY KEY(PURCHASE_REQUEST_ID,PURCHASE_REQUEST_LINE_ITEM_D) | | |

| REIMBURSEMENT_REQUEST_LINE_ITEMS | | |
|--------------------------------------------------------------------------|--------------|--------------------------------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK,FK,NULLABLE) |
| REIMBURSEMENT_REQUEST_ID | INTEGER | FOREIGN KEY(REIMBURSEMENT REQUEST -> REIMBURSEMENT_REQUEST_ID), NOT NULL |
| REIMBURSEMENT_REQUEST_LINE_ITEM_ID | INTEGER | NOT NULL |
| ITEM_DESCRIPTION | VARCHAR(50) | NOT NULL |
| ITEM_AMOUNT | DECIMAL(7,2) | NOT NULL |
| PRIMARY KEY (REIMBURSEMENT_REQUEST_ID,REIMBURSEMENT_REQUEST_LINE_ITEM_D) | | |

| PURCHASE_ORDER | | |
|------------------------|--------------|----------------------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK,FK,NULLABLE) |
| ORDER_ID | INTEGER | PRIMARY KEY, NOT NULL |
| PURCHASE_REQUEST_ID | INTEGER | FOREIGN KEY(PURCHASE_REQUEST -> PURCHASE_REQUEST_ID), NOT NULL |
| ORDER_CREATED_DATE | DATETIME | NOT NULL |
| ORDER_STATUS | VARCHAR(20) | NOT NULL, DOMAIN VALUES - (OPEN, WIP, FULFILLED, COMPLETED) |
| ORDER_FULFILLMENT_DATE | DATETIME | NULL |
| CREATED_BY | INTEGER | NOT NULL |
| ASSIGNED_TO | INTEGER | FOREIGN KEY (SUPPLIER -> SUPPLIER_ID), NULL |
| DEPARTMENT | VARCHAR(20) | NOT NULL |
| WORK_LOCATION | VARCHAR (20) | NOT NULL |
| ORDER_COMPLETED_DATE | DATETIME | NULL |

| INVOICE | | |
|------------------|---------------|---------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK,FK,NULLABLE) |
| INVOICE_ID | INTEGER | PRIMARY KEY, NOT NULL |
| ORDER_ID | INTEGER | FOREIGN KEY(PURCHASE_ORDER -> ORDER_ID), NOT NULL |
| SUPPLIER_ID | INTEGER | FOREIGN KEY(SUPPLIER -> SUPPLIER_ID), NOT NULL |
| INVOICE_AMOUNT | DECIMAL(10,2) | NOT NULL |
| PAYMENT_DUE_DATE | DATETIME | NOT NULL |

| INVOICE_LINE_ITEMS | | |
|----------------------------------------------|---------------|------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK,FK,NULLABLE) |
| INVOICE_ID | INTEGER | FOREIGN KEY (INVOICE --> INVOICE_ID), NOT NULL |
| INVOICE_LINE_ITEM_ID | INTEGER | NOT NULL |
| ITEM_DESCRIPTION | VARCHAR(50) | NOT NULL |
| ITEM_QUANTITY_ORDERED | INTEGER | NOT NULL |
| ITEM_QUANTITY_DELIVERED | INTEGER | NOT NULL |
| ITEM_PRICE | DECIMAL(7,2) | NOT NULL |
| TOTAL_ITEM_PRICE | DECIMAL(10,2) | NOT NULL |
| PRIMARY KEY(INVOICE_ID,INVOICE_LINE_ITEM_ID) | | |

| INVOICE_PAYMENT | | |
|------------------------|---------------|----------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT(PK,FK,NULLABLE) |
| INVOICE_PAYMENT_ID | INTEGER | PRIMARY KEY, NOT NULL |
| INVOICE_PAYMENT_AMOUNT | DECIMAL(10,2) | NOT NULL |
| INVOICE_PAYMENT_DATE | DATETIME | NOT NULL |
| INVOICE_ID | INTEGER | FOREIGN KEY(INVOICE -> INVOICE_ID), NOT NULL |
| PAID_BY | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NULL |

| REIMBURSEMENT_PAYMENT | | |
|------------------------------|--------------|---------------------------------------------------------------------------|
| COLUMN_NAME | DATA_TYPE | CONSTRAINT (PK, FK, NULLABLE) |
| REIMBURSEMENT_PAYMENT_ID | INTEGER | PRIMARY KEY, NOT NULL |
| REIMBURSEMENT_PAYMENT_AMOUNT | DECIMAL(7,2) | NOT NULL |
| REIMBURSEMENT_PAYMENT_DATE | DATETIME | NOT NULL |
| REIMBURSEMENT_REQUEST_ID | INTEGER | FOREIGN KEY (REIMBURSEMENT REQUEST -> REIMBURSEMENT REQUEST_ID), NOT NULL |
| PAID_BY | INTEGER | FOREIGN KEY (EMPLOYEE -> EMPLOYEE_ID), NOT NULL |

6. SQL Scripts

- Initial state script – Contains creation of tables and population of data
- Final script – Contains all inserts, updates and deletions

Both scripts submitted along with Report

7. SQL queries

Business Process 1:

When a new employee joins the organization his/her details are inserted in the employee table. Any changes to these details would be updated as and when needed. When the employee leaves the organization this record is deleted.

Insert a new employee

Step 1: Insert employee into Employee table

BEFORE PROCESSING THE INSERT EMPLOYEE QUERY

The screenshot shows the SQL Server Management Studio interface. The query window contains the following SQL code:

```
SELECT*
FROM EMPLOYEE
```

The results grid displays 28 rows of employee data. The columns are:

| | EMPLOYEE_ID | EMPLOYEE_NAME | EMAIL_ID | CONTACT_PHONE | DESIGNATION | DEPARTMENT | WORK_LOCATION | EXPENSE_LIMIT | APPROVAL_LIMIT |
|----|-------------|-----------------|-----------------------|---------------|--------------|------------|---------------|---------------|----------------|
| 1 | 1000 | John Hanks | jhanks@riotgames.com | 408-872-2919 | Associate | Marketing | Santa Clara | 5000.00 | NULL |
| 2 | 1001 | Meera M | meeram@riotgames.com | 408-467-2810 | Manager | Technology | Milpitas | 15000.00 | 20000.00 |
| 3 | 1002 | Rohit P | rohitp@riotgames.com | 916-693-2811 | Associate | Sales | New York | 5000.00 | NULL |
| 4 | 1003 | Frank Mank | frankm@riotgames.com | 607-727-6597 | Manager | Technology | Santa Clara | 15000.00 | 20000.00 |
| 5 | 1004 | Jade Turel | jadet@riotgames.com | 408-467-2813 | Associate | Sales | Milpitas | 5000.00 | NULL |
| 6 | 1005 | Vanita Pandey | vanitap@riotgames.com | 408-467-2814 | Associate | Technology | Milpitas | 5000.00 | NULL |
| 7 | 1006 | Rishab Apte | rishaba@riotgames.com | 607-565-6665 | Associate | Accounting | New York | 5000.00 | NULL |
| 8 | 1007 | Robert McCarrie | robertm@riotgames.com | 408-467-2816 | Associate | Technology | New York | 5000.00 | NULL |
| 9 | 1008 | Ashly John | ashlyj@riotgames.com | 607-727-6595 | Manager | Accounting | San Francisco | 15000.00 | 20000.00 |
| 10 | 1009 | Jon Hansen | jonh@riotgames.com | 916-467-2913 | Associate | Production | Milpitas | 5000.00 | NULL |
| 11 | 1010 | Nelly Geifman | nellyg@riotgames.com | 916-364-2915 | Sr Analyst | Technology | New York | 10000.00 | NULL |
| 12 | 1011 | Rohit Bhaita | rohitbh@riotgames.com | 408-467-2820 | Manager | Technology | Milpitas | 15000.00 | 20000.00 |
| 13 | 1012 | Lisa Bob | lisab@riotgames.com | 916-467-2821 | Associate | Sales | Milpitas | 5000.00 | NULL |
| 14 | 1013 | Alice Joshten | alicej@riotgames.com | 408-981-4983 | Associate | Technology | San Francisco | 5000.00 | NULL |
| 15 | 1014 | Ash Groven | ashg@riotgames.com | 916-767-2888 | Sr Analyst | Sales | Santa Clara | 10000.00 | NULL |
| 16 | 1015 | Neha Goyal | nehang@riotgames.com | 408-411-2210 | Associate | Technology | New York | 5000.00 | NULL |
| 17 | 1016 | Taylor G | taylorg@riotgames.com | 607-761-1132 | Sr Associate | Accounting | San Diego | 10000.00 | NULL |
| 18 | 1017 | Simon K | simonk@riotgames.com | 916-364-2123 | Analyst | Admin | San Francisco | 10000.00 | NULL |
| 19 | 1018 | Darrel Graph | darrelg@riotgames.com | 607-737-2595 | Associate | Marketing | San Diego | 5000.00 | NULL |
| 20 | 1019 | Rinky K | rinkyk@riotgames.com | 408-466-2800 | Manager | Marketing | San Diego | 15000.00 | 20000.00 |
| 21 | 1020 | Lossie Pace | lossiep@riotgames.com | 916-365-2124 | Associate | Admin | San Francisco | 5000.00 | NULL |
| 22 | 1021 | Belle Swan | bellas@riotgames.com | 408-678-900 | Sr Manager | Technology | Milpitas | 30000.00 | 50000.00 |
| 23 | 1022 | Edward Cullen | edwardc@riotgames.com | 775-898-234 | Sr Manager | Accounting | San Francisco | 30000.00 | 50000.00 |
| 24 | 1023 | Alice Graham | aliceg@riotgames.com | 916-101-2345 | Sr Manager | Technology | Milpitas | 30000.00 | 50000.00 |
| 25 | 1024 | Xiu Zhang | xiz@riotgames.com | 607-889-2456 | Manager | Production | Milpitas | 15000.00 | 20000.00 |
| 26 | 1025 | Akshay S | Akshays@riotgames.com | 510-245-9876 | Manager | Sales | Santa Clara | 15000.00 | 20000.00 |
| 27 | 1026 | Ping Zhe | pingz@riotgames.com | 510-454-8080 | Manager | Admin | San Francisco | 15000.00 | 20000.00 |
| 28 | 1027 | Jack Nicholas | jackn@riotgames.com | 408-929-3456 | Associate | Accounting | San Francisco | 5000.00 | NULL |

Query executed successfully. | RISHABH\SQL EXPRESS (12.0 RTM)

Ln 4

QUERY PROCESSING

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, there is a connection to 'RISHABH\SQLEXPRESS (SQL Server 12.0.0)'. In the center pane, a query window titled 'SQLQuery17.sql' is open, containing the following SQL code:

```
INSERT INTO EMPLOYEE (EMPLOYEE_ID, EMPLOYEE_NAME, EMAIL_ID, CONTACT_NO, DESIGNATION, DEPARTMENT, WORK_LOCATION, EXPENSE_LIMIT)
VALUES ('1028', 'Jiminy Jade', 'jiminy@notgames.com', '408-208-9198', 'Associate', 'Accounting', 'New York', '5000')
```

Below the code, the message '(1 row(s) affected)' is displayed. At the bottom of the window, a yellow status bar says 'Query executed successfully.' To the right, the taskbar shows the system status: RISHABH\SQLEXPRESS (12.0 RTM) - RISHABH\Rishabh (53) master 00:00:00 0 rows.

AFTER PROCESSING THE INSERT EMPLOYEE QUERY

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, there is a connection to 'RISHABH\SQLEXPRESS (SQL Server 12.0.0)'. In the center pane, a query window titled 'SQLQuery17.sql' is open, containing the following SQL code:

```
select * from EMPLOYEE
```

The results grid displays 29 rows of employee data. The columns are: EMPLOYEE_ID, EMPLOYEE_NAME, EMAIL_ID, CONTACT_NO, DESIGNATION, DEPARTMENT, WORK_LOCATION, EXPENSE_LIMIT, APPROVAL_LIMIT. The data includes various employees like John Henika, Meera M, Rohit P, Frank Mank, Jade Turel, Venita Pandey, Rishab Apte, Robert McCann, Ashly John, Jon Hensen, Nelly Gefman, Rohit Bhate, Lisa Boli, Alice Joshua, Ash Grover, Neha Goval, Taylor G, Simon K, Daniel Graph, Ricky K, Lossia Peice, Bulte Swan, Edward Cullen, Alice Graham, Xu Zheng, Akshay S, Ping Zha, Jack Nicholas, and Jiminy Jade, each with their respective details such as email addresses, contact numbers, and department assignments.

At the bottom of the window, a yellow status bar says 'Query executed successfully.' To the right, the taskbar shows the system status: RISHABH\SQLEXPRESS (12.0 RTM) - RISHABH\Rishabh (53) master 00:00:00 29 rows.

Step 2: Insert Employee record 1028 in MANAGES

BEFORE PROCESSING THE QUERY

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery17.sql - RISHABH\SQLEXPRESS.master (RISHABH\Rishabh (53))' is open. The query is:

```
INSERT INTO MANAGES (EMPLOYEE_ID, MANAGER_ID)
VALUES ('1027', '1008')

SELECT
FROM MANAGES
```

The results pane displays a table with columns 'EMPLOYEE_ID' and 'MANAGER_ID'. The data is as follows:

| EMPLOYEE_ID | MANAGER_ID |
|-------------|------------|
| 1 | 1000 |
| 2 | 1001 |
| 3 | 1002 |
| 4 | 1003 |
| 5 | 1004 |
| 6 | 1005 |
| 7 | 1006 |
| 8 | 1007 |
| 9 | 1008 |
| 10 | 1009 |
| 11 | 1010 |
| 12 | 1011 |
| 13 | 1012 |
| 14 | 1013 |
| 15 | 1014 |
| 16 | 1015 |
| 17 | 1016 |
| 18 | 1017 |
| 19 | 1018 |
| 20 | 1020 |
| 21 | 1027 |

A message at the bottom of the results pane says 'Query executed successfully.' The status bar at the bottom right shows 'RISHABH\SQLEXPRESS (12.0 RTM) RISHABH\Rishabh (53) master 00:00:00 - 21 rows'.

QUERY PROCESSING & AFTER PROCESSING THE QUERY

The screenshot shows the Microsoft SQL Server Management Studio interface after the query has been processed. The query window is identical to the previous one:

```
INSERT INTO MANAGES (EMPLOYEE_ID, MANAGER_ID)
VALUES ('1028', '1008')

SELECT
FROM MANAGES
```

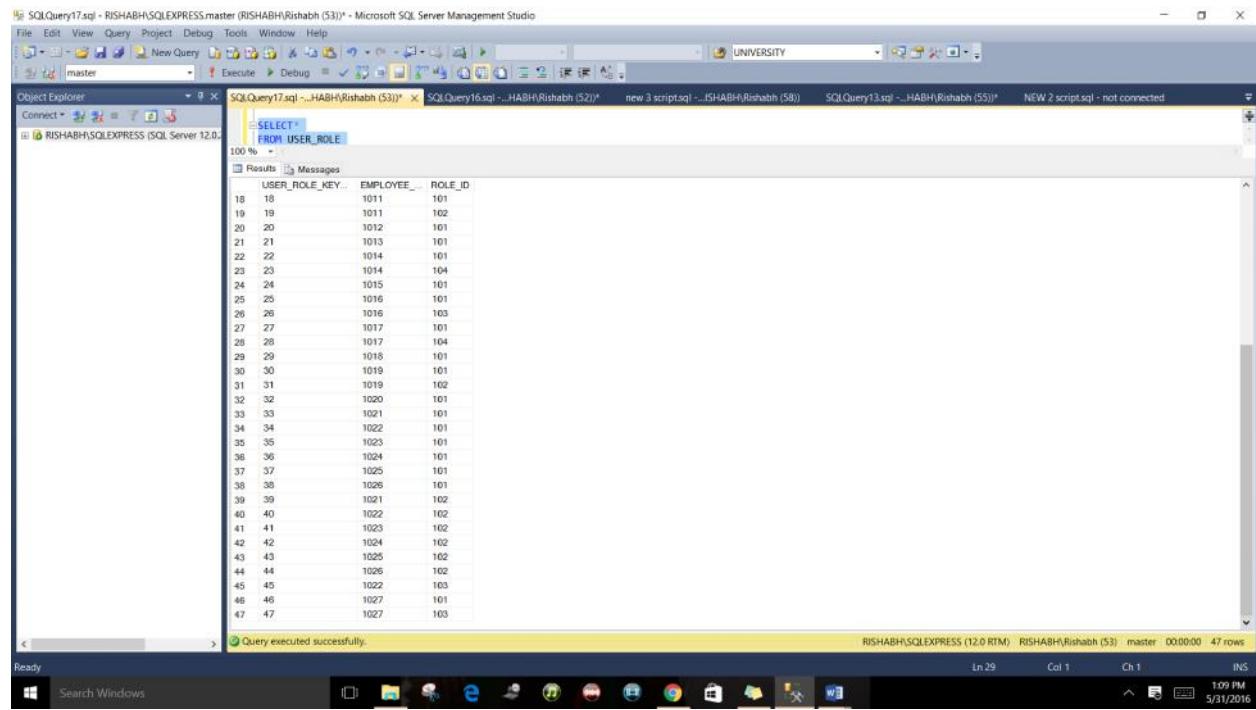
The results pane displays a table with columns 'EMPLOYEE_ID' and 'MANAGER_ID'. The data is as follows:

| EMPLOYEE_ID | MANAGER_ID |
|-------------|------------|
| 1 | 1000 |
| 2 | 1001 |
| 3 | 1002 |
| 4 | 1003 |
| 5 | 1004 |
| 6 | 1005 |
| 7 | 1006 |
| 8 | 1007 |
| 9 | 1008 |
| 10 | 1009 |
| 11 | 1010 |
| 12 | 1011 |
| 13 | 1012 |
| 14 | 1013 |
| 15 | 1014 |
| 16 | 1015 |
| 17 | 1016 |
| 18 | 1017 |
| 19 | 1018 |
| 20 | 1020 |
| 21 | 1027 |
| 22 | 1028 |

A message at the bottom of the results pane says 'Query executed successfully.' The status bar at the bottom right shows 'RISHABH\SQLEXPRESS (12.0 RTM) RISHABH\Rishabh (53) master 00:00:00 - 22 rows'.

Step 3: Insert Employee roles in USER_ROLE

BEFORE PROCESSING THE INSERT QUERY FOR USER_ROLE



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window displays the following SELECT statement:

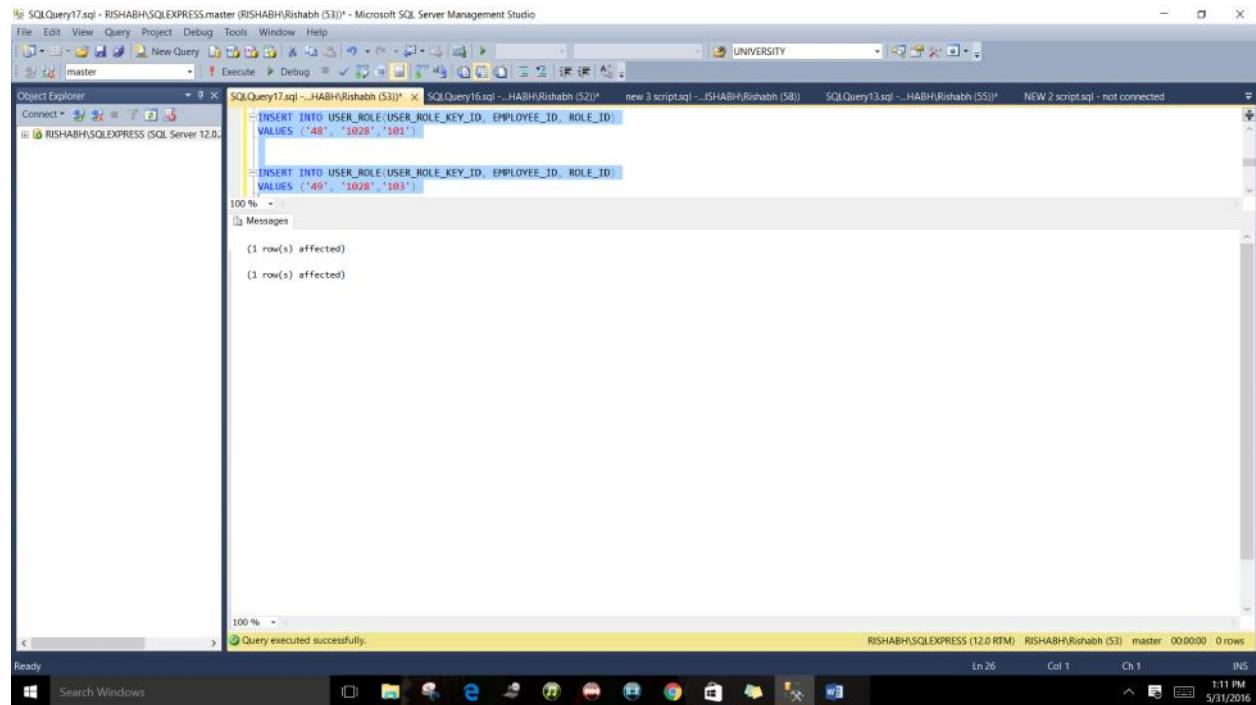
```
SELECT * FROM USER_ROLE
```

The results grid shows 47 rows of data from the USER_ROLE table:

| USER_ROLE_KEY_ID | EMPLOYEE_ID | ROLE_ID |
|------------------|-------------|---------|
| 18 | 18 | 101 |
| 19 | 19 | 102 |
| 20 | 20 | 1012 |
| 21 | 21 | 1013 |
| 22 | 22 | 1014 |
| 23 | 23 | 1014 |
| 24 | 24 | 1015 |
| 25 | 25 | 1016 |
| 26 | 26 | 1016 |
| 27 | 27 | 1017 |
| 28 | 28 | 1017 |
| 29 | 29 | 1018 |
| 30 | 30 | 1019 |
| 31 | 31 | 1019 |
| 32 | 32 | 1020 |
| 33 | 33 | 1021 |
| 34 | 34 | 1022 |
| 35 | 35 | 1023 |
| 36 | 36 | 1024 |
| 37 | 37 | 1025 |
| 38 | 38 | 1026 |
| 39 | 39 | 1021 |
| 40 | 40 | 1022 |
| 41 | 41 | 1023 |
| 42 | 42 | 1024 |
| 43 | 43 | 1025 |
| 44 | 44 | 1026 |
| 45 | 45 | 1022 |
| 46 | 46 | 1027 |
| 47 | 47 | 1027 |

At the bottom of the results grid, a message indicates: "Query executed successfully."

QUERY PROCESSING



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window displays the following two INSERT statements:

```
INSERT INTO USER_ROLE(USER_ROLE_KEY_ID, EMPLOYEE_ID, ROLE_ID)  
VALUES ('48', '1028', '101')
```

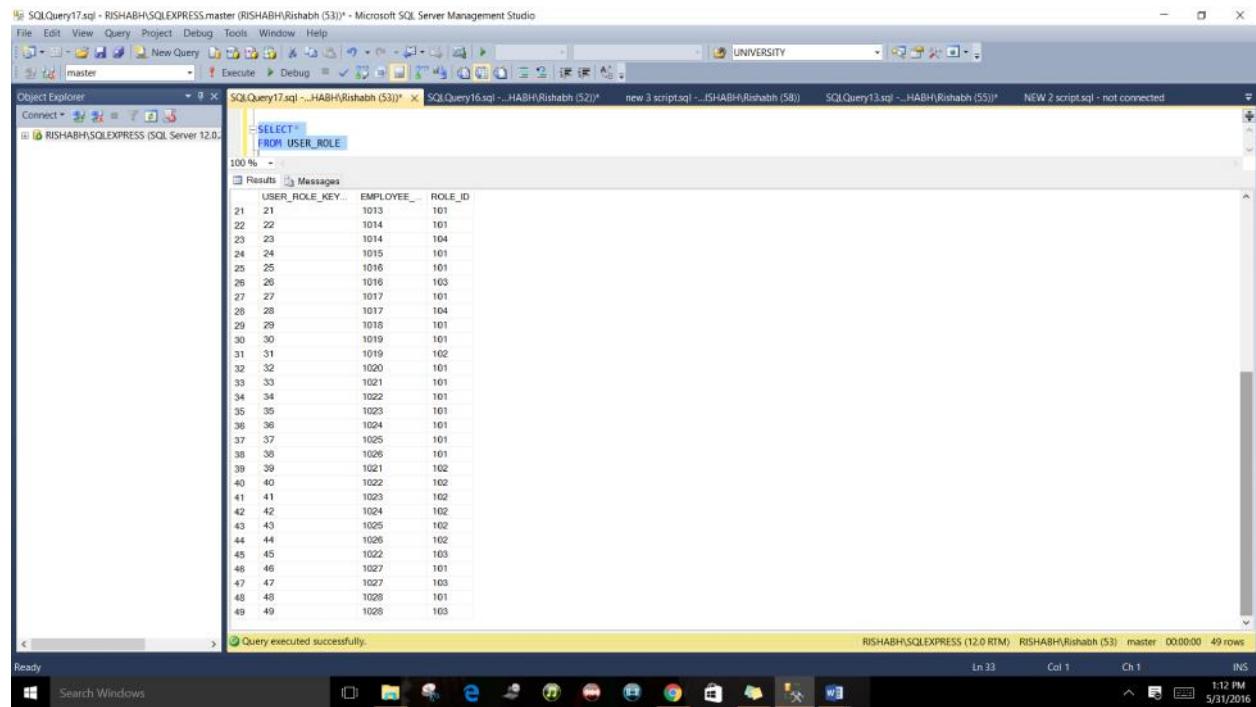
```
INSERT INTO USER_ROLE(USER_ROLE_KEY_ID, EMPLOYEE_ID, ROLE_ID)  
VALUES ('49', '1028', '103')
```

The results grid shows the affected rows:

| MESSAGE |
|---------------------|
| (1 row(s) affected) |
| (1 row(s) affected) |

At the bottom of the results grid, a message indicates: "Query executed successfully."

AFTER PROCESSING THE QUERY



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window is open with the following SQL code:

```
SELECT * FROM USER_ROLE
```

The results grid displays 49 rows of data from the USER_ROLE table, with columns labeled USER_ROLE_KEY, EMPLOYEE_ID, and ROLE_ID. The data is as follows:

| | USER_ROLE_KEY | EMPLOYEE_ID | ROLE_ID |
|----|---------------|-------------|---------|
| 21 | 21 | 1013 | 101 |
| 22 | 22 | 1014 | 101 |
| 23 | 23 | 1014 | 104 |
| 24 | 24 | 1015 | 101 |
| 25 | 25 | 1016 | 101 |
| 26 | 26 | 1016 | 103 |
| 27 | 27 | 1017 | 101 |
| 28 | 28 | 1017 | 104 |
| 29 | 29 | 1018 | 101 |
| 30 | 30 | 1019 | 101 |
| 31 | 31 | 1019 | 102 |
| 32 | 32 | 1020 | 101 |
| 33 | 33 | 1021 | 101 |
| 34 | 34 | 1022 | 101 |
| 35 | 35 | 1023 | 101 |
| 36 | 36 | 1024 | 101 |
| 37 | 37 | 1025 | 101 |
| 38 | 38 | 1026 | 101 |
| 39 | 39 | 1021 | 102 |
| 40 | 40 | 1022 | 102 |
| 41 | 41 | 1023 | 102 |
| 42 | 42 | 1024 | 102 |
| 43 | 43 | 1025 | 102 |
| 44 | 44 | 1026 | 102 |
| 45 | 45 | 1022 | 103 |
| 46 | 46 | 1027 | 101 |
| 47 | 47 | 1027 | 103 |
| 48 | 48 | 1028 | 101 |
| 49 | 49 | 1028 | 103 |

Below the results, a message bar indicates "Query executed successfully." The status bar at the bottom right shows the session details: RISHABH\SQLEXPRESS (12.0 RTM) | RISHABH\Rishabh (53) | master | 0000:00 | 49 rows.

Update employee contact number

```
UPDATE EMPLOYEE  
SET EMPLOYEE_NAME = 'Meera Seth', CONTACT_NO = '898-206-797',  
EMAIL_ID = 'meeras@riotgames.com'  
FROM EMPLOYEE E  
WHERE E.EMPLOYEE_ID = '1001'
```

BEFORE EXECUTING THE QUERY

S-2603-Project-Expensify (RISHABH\Rishabh (59))* - Microsoft SQL Server Management Studio

Tools Window Help

Execute Debug SQLQuery6.sql - RI...HABH\Rishabh (60)* SQLQuery5.sql - RI...HABH\Rishabh (59)* SQLQuery4.sql - RI...HABH\Rishabh (58)* SQLQuery3.sql - RI...HABH\Rishabh (55)

```
select*
from EMPLOYEE
```

100 %

Results Messages

| | EMPLOYEE_ID | EMPLOYEE_NA... | EMAIL_ID | CONTACT_... | DESIGNATI... | DEPARTME... | WORK_LOCATI... | EXPENSE_LIMIT | APPROVAL_LIMIT |
|----|-------------|-----------------|-----------------------|--------------|--------------|-------------|----------------|---------------|----------------|
| 1 | 1000 | John Hanks | jhanks@riotgames.com | 408-872-2919 | Associate | Marketing | Santa Clara | 5000.00 | NULL |
| 2 | 1001 | Meera M | meeram@riotgames.com | 408-467-2810 | Manager | Technology | Milpitas | 15000.00 | 20000.00 |
| 3 | 1002 | Rohit P | rohitp@riotgames.com | 916-693-2811 | Associate | Sales | New York | 5000.00 | NULL |
| 4 | 1003 | Frank Mank | frankm@riotgames.com | 607-727-6597 | Manager | Technology | Santa Clara | 15000.00 | 20000.00 |
| 5 | 1004 | Jade Turel | jedett@riotgames.com | 408-467-2813 | Associate | Sales | Milpitas | 5000.00 | NULL |
| 6 | 1005 | Vanita Pandey | vanitap@riotgames.com | 408-467-2814 | Associate | Technology | Milpitas | 5000.00 | NULL |
| 7 | 1006 | Rishab Apte | rishab@riotgames.com | 607-585-6665 | Associate | Accounting | New York | 5000.00 | NULL |
| 8 | 1007 | Robert McCarrie | robertm@riotgames.com | 408-467-2816 | Associate | Technology | New York | 5000.00 | NULL |
| 9 | 1008 | Ashly John | ashly@riotgames.com | 607-727-6595 | Manager | Accounting | San Francisco | 15000.00 | 20000.00 |
| 10 | 1009 | Jon Hansen | jonh@riotgames.com | 916-467-2913 | Associate | Production | Milpitas | 5000.00 | NULL |
| 11 | 1010 | Nelly Geifman | nellyg@riotgames.com | 916-364-2915 | Sr Analyst | Technology | New York | 10000.00 | NULL |
| 12 | 1011 | Rohit Bhatia | rohitbh@riotgames.com | 408-467-2820 | Manager | Technology | Milpitas | 15000.00 | 20000.00 |
| 13 | 1012 | Lisa Bob | lisab@riotgames.com | 916-467-2821 | Associate | Sales | Milpitas | 5000.00 | NULL |
| 14 | 1013 | Alice Joshten | alicej@riotgames.com | 408-981-4983 | Associate | Technology | San Francisco | 5000.00 | NULL |
| 15 | 1014 | Ash Groven | ashg@riotgames.com | 916-767-2888 | Sr Analyst | Sales | Santa Clara | 10000.00 | NULL |

Query executed successfully.

RISHABH\SQLEXPRESS (12.0 RTM) | RISHABH\Rishab

Ln 2

AFTER EXECUTING THE QUERY

The screenshot shows the SQL Server Management Studio interface with four tabs at the top: SQLQuery7.sql - RI...HABH\Rishabh (61)*, SQLQuery6.sql - RI...HABH\Rishabh (60)*, SQLQuery5.sql - RI...HABH\Rishabh (59)*, and SQLQuery4.sql - RI...HABH\Rishabh (58)*. The first tab contains the executed query:

```
SELECT *
FROM EMPLOYEE
```

The results pane displays the data from the EMPLOYEE table:

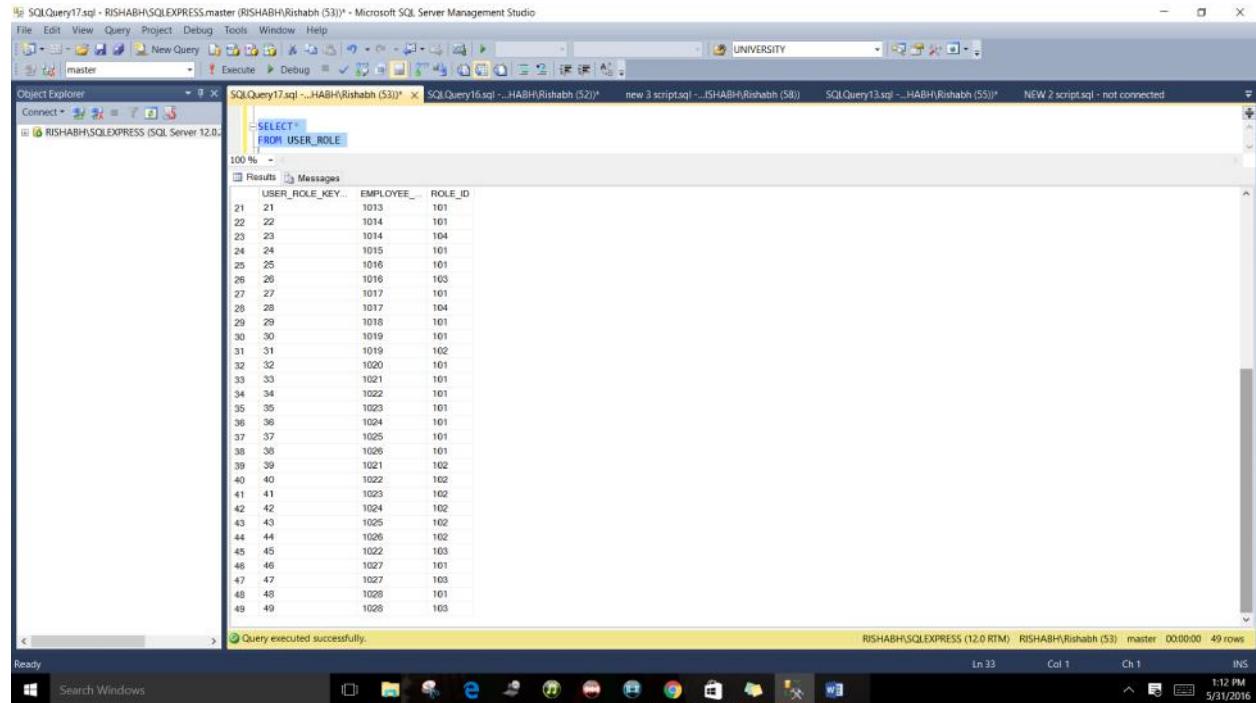
| | EMPLOYEE_ID | EMPLOYEE_NAME | EMAIL_ID | CONTACT_PHONE | DESIGNATION | DEPARTMENT | WORK_LOCATION | EXPENSE_LIMIT | APPROVAL_LIMIT |
|----|-------------|-----------------|------------------------|---------------|-------------|------------|---------------|---------------|----------------|
| 1 | 1000 | John Hanks | jhanks@riotgames.com | 408-872-2919 | Associate | Marketing | Santa Clara | 5000.00 | NULL |
| 2 | 1001 | Meera Seth | meeras@riotgames.com | 898-206-797 | Manager | Technology | Milpitas | 15000.00 | 20000.00 |
| 3 | 1002 | Rohit P | rohitp@riotgames.com | 916-693-2811 | Associate | Sales | New York | 5000.00 | NULL |
| 4 | 1003 | Frank Mank | frankm@riotgames.com | 607-727-6597 | Manager | Technology | Santa Clara | 15000.00 | 20000.00 |
| 5 | 1004 | Jade Turel | jadet@riotgames.com | 408-467-2813 | Associate | Sales | Milpitas | 5000.00 | NULL |
| 6 | 1005 | Vanita Pandey | vanitap@riotgames.com | 408-467-2814 | Associate | Technology | Milpitas | 5000.00 | NULL |
| 7 | 1006 | Rishabh Apte | rishabha@riotgames.com | 607-565-6665 | Associate | Accounting | New York | 5000.00 | NULL |
| 8 | 1007 | Robert McCarrie | robertm@riotgames.com | 408-467-2816 | Associate | Technology | New York | 5000.00 | NULL |
| 9 | 1008 | Ashly John | ashlyj@riotgames.com | 607-727-6595 | Manager | Accounting | San Francisco | 15000.00 | 20000.00 |
| 10 | 1009 | Jon Hansen | jonh@riotgames.com | 916-467-2913 | Associate | Production | Milpitas | 5000.00 | NULL |
| 11 | 1010 | Nelly Geifman | nellyg@riotgames.com | 916-364-2915 | Sr Analyst | Technology | New York | 10000.00 | NULL |
| 12 | 1011 | Rohit Bhatia | rohitbh@riotgames.com | 408-467-2820 | Manager | Technology | Milpitas | 15000.00 | 20000.00 |
| 13 | 1012 | Lisa Bob | lisab@riotgames.com | 916-467-2821 | Associate | Sales | Milpitas | 5000.00 | NULL |
| 14 | 1013 | Alice Joshten | alicej@riotgames.com | 408-981-4983 | Associate | Technology | San Francisco | 5000.00 | NULL |
| 15 | 1014 | Ash Groven | ashg@riotgames.com | 916-767-2888 | Sr Analyst | Sales | Santa Clara | 10000.00 | NULL |

At the bottom left, it says "Query executed successfully." and at the bottom right, it says "RISHABH\SOLEXPRESS (12.0 RTM) | RISHABH\Rish".

Delete an employee

Step 1: Delete user role from USER_ROLE for EMPLOYEE ID 1028

BEFORE PROCESSING THE QUERY



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'RISHABH\SQLEXPRESS' is selected. In the center pane, a query window displays the following SQL code:

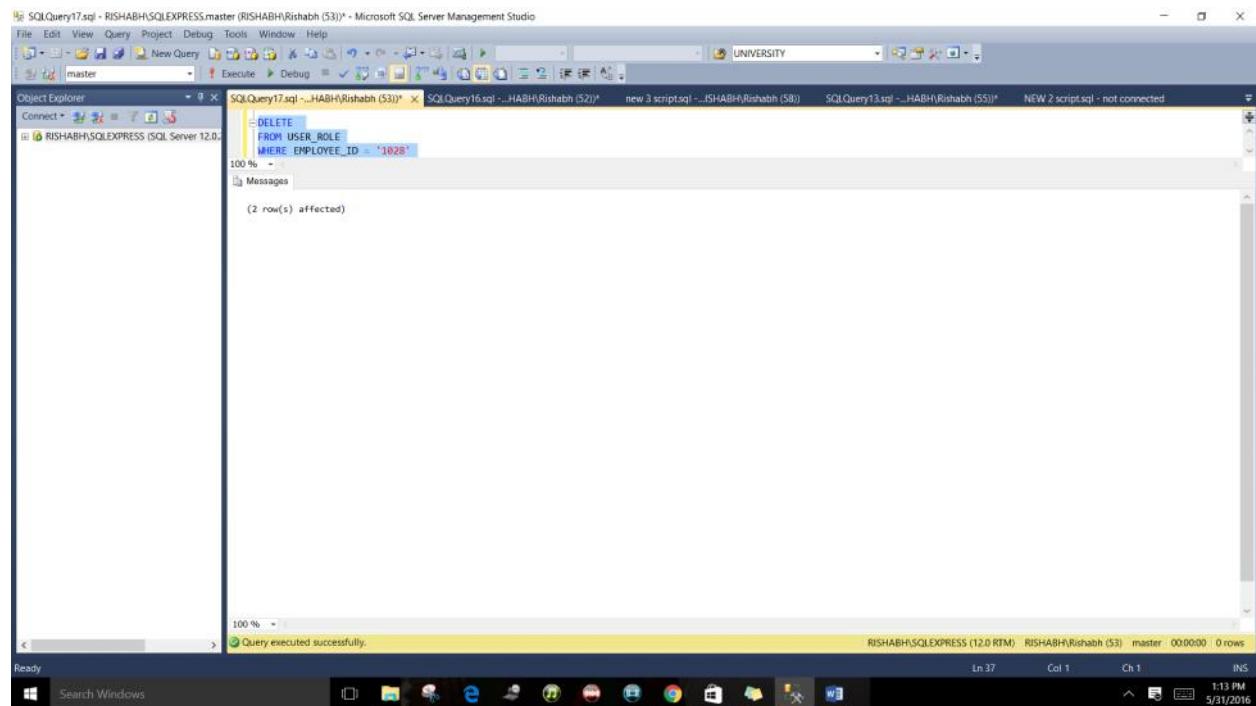
```
SELECT * FROM USER_ROLE
```

The results grid shows the following data:

| USER_ROLE_KEY | EMPLOYEE_ID | ROLE_ID |
|---------------|-------------|---------|
| 21 | 21 | 1013 |
| 22 | 22 | 1014 |
| 23 | 23 | 1014 |
| 24 | 24 | 1015 |
| 25 | 25 | 1016 |
| 26 | 26 | 1016 |
| 27 | 27 | 1017 |
| 28 | 28 | 1017 |
| 29 | 29 | 1018 |
| 30 | 30 | 1019 |
| 31 | 31 | 1019 |
| 32 | 32 | 1020 |
| 33 | 33 | 1021 |
| 34 | 34 | 1022 |
| 35 | 35 | 1023 |
| 36 | 36 | 1024 |
| 37 | 37 | 1025 |
| 38 | 38 | 1026 |
| 39 | 39 | 1021 |
| 40 | 40 | 1022 |
| 41 | 41 | 1023 |
| 42 | 42 | 1024 |
| 43 | 43 | 1025 |
| 44 | 44 | 1026 |
| 45 | 45 | 1022 |
| 46 | 46 | 1027 |
| 47 | 47 | 1027 |
| 48 | 48 | 1028 |
| 49 | 49 | 1028 |

At the bottom of the results grid, a message states: "Query executed successfully." The status bar at the bottom right shows: RISHABH\SQLEXPRESS (12.0 RTM) - RISHABH\Rishabh (53) master : 000000 - 49 rows.

QUERY PROCESSING



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'RISHABH\SQLEXPRESS' is selected. In the center pane, a query window displays the following SQL code:

```
DELETE FROM USER_ROLE WHERE Employee_ID = '1028'
```

The results grid shows the message: "(2 row(s) affected)". At the bottom of the results grid, a message states: "Query executed successfully." The status bar at the bottom right shows: RISHABH\SQLEXPRESS (12.0 RTM) - RISHABH\Rishabh (53) master : 000000 - 0 rows.

AFTER PROCESSING THE QUERY

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window is open with the following SQL code:

```
SELECT * FROM USER_ROLE
```

The results grid displays 47 rows of data from the USER_ROLE table, showing columns: USER_ROLE_KEY, EMPLOYEE_ID, and ROLE_ID. The data includes various employee IDs mapped to different roles.

At the bottom of the screen, a message bar indicates "Query executed successfully." and shows the execution details: RISHABH\SQLEXPRESS (12.0 RTM) - RISHABH\Rishabh (53) master 00:00:00 47 rows.

Step 2: Delete from MANAGES table

BEFORE PROCESSING THE QUERY

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window is open with the following SQL code:

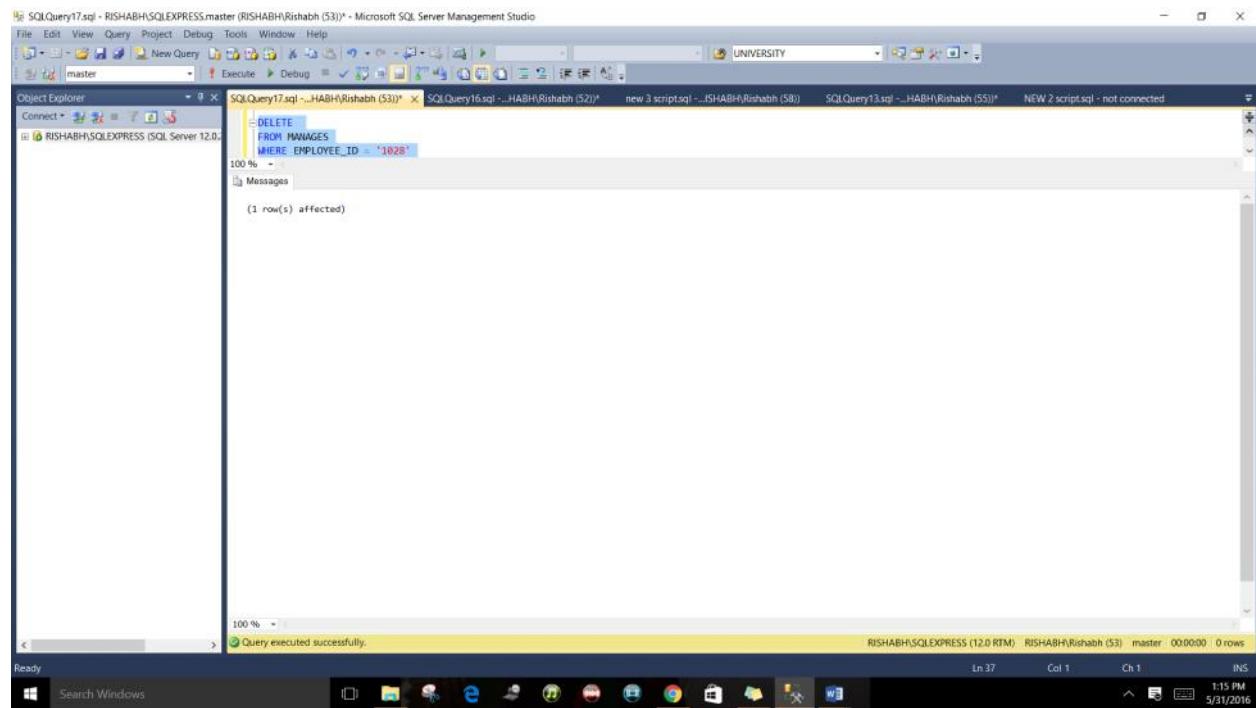
```
INSERT INTO MANAGES (EMPLOYEE_ID, MANAGER_ID)
VALUES ('1028', '1008')

SELECT * FROM MANAGES
```

The results grid displays 22 rows of data from the MANAGES table, showing columns: EMPLOYEE_ID and MANAGER_ID. The data includes various employee IDs mapped to their managers.

At the bottom of the screen, a message bar indicates "Query executed successfully." and shows the execution details: RISHABH\SQLEXPRESS (12.0 RTM) - RISHABH\Rishabh (53) master 00:00:00 22 rows.

QUERY PROCESSING



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery17.sql' is open, displaying the following SQL code:

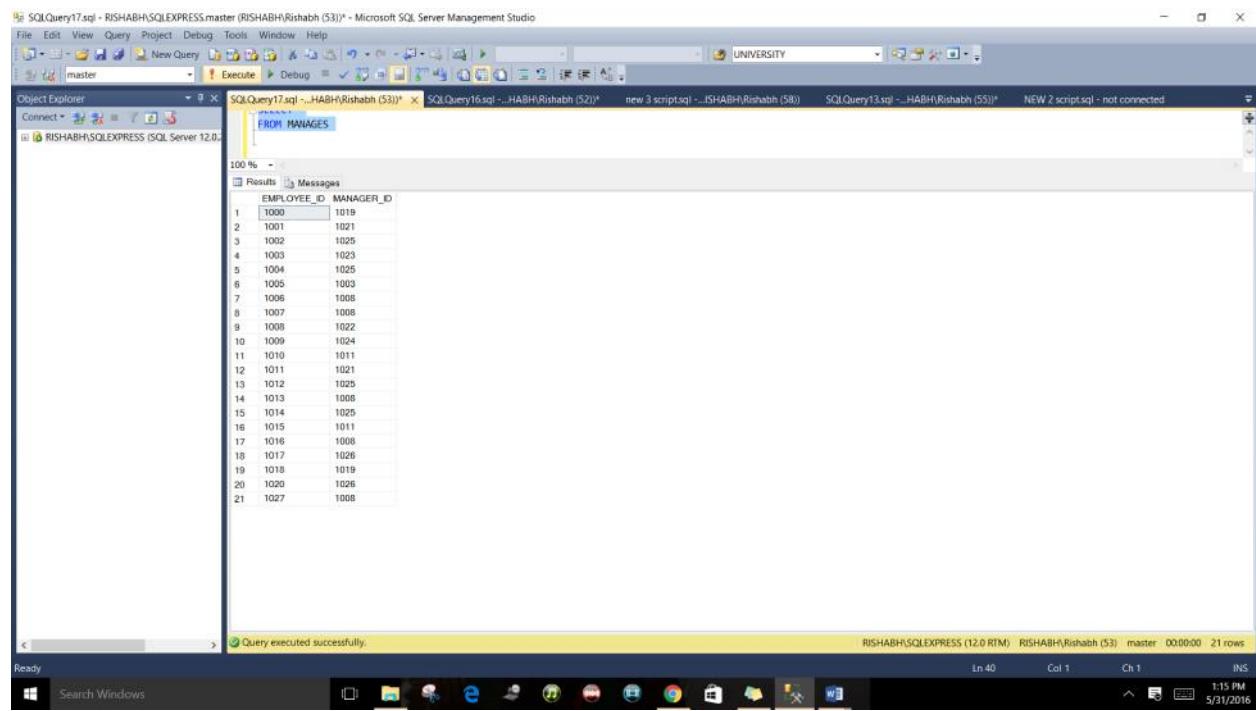
```
DELETE
FROM MANAGES
WHERE EMPLOYEE_ID = '1028'
```

The results pane shows the output of the query:

```
(1 row(s) affected)
```

A status bar at the bottom indicates 'Query executed successfully.' and provides session details: RISHABH\SQLEXPRESS (12.0 RTM) - RISHABH\Rishabh (53) - master - 00:00:00 - 0 rows.

AFTER PROCESSING THE QUERY



The screenshot shows the Microsoft SQL Server Management Studio interface after the query has been processed. A query window titled 'SQLQuery17.sql' is open, displaying the following SQL code:

```
SELECT * FROM MANAGES
```

The results pane shows the output of the query, which is a list of employee IDs and their manager IDs:

| | EMPLOYEE_ID | MANAGER_ID |
|----|-------------|------------|
| 1 | 1000 | 1019 |
| 2 | 1001 | 1021 |
| 3 | 1002 | 1025 |
| 4 | 1003 | 1023 |
| 5 | 1004 | 1025 |
| 6 | 1005 | 1003 |
| 7 | 1006 | 1008 |
| 8 | 1007 | 1008 |
| 9 | 1008 | 1022 |
| 10 | 1009 | 1024 |
| 11 | 1010 | 1011 |
| 12 | 1011 | 1021 |
| 13 | 1012 | 1025 |
| 14 | 1013 | 1008 |
| 15 | 1014 | 1025 |
| 16 | 1015 | 1011 |
| 17 | 1016 | 1008 |
| 18 | 1017 | 1026 |
| 19 | 1018 | 1019 |
| 20 | 1020 | 1026 |
| 21 | 1027 | 1008 |

A status bar at the bottom indicates 'Query executed successfully.' and provides session details: RISHABH\SQLEXPRESS (12.0 RTM) - RISHABH\Rishabh (53) - master - 00:00:00 - 21 rows.

Step 3: Delete employee record from Employee table

BEFORE PROCESSING THE QUERY

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer pane shows a connection to 'RISHABH\SOLEXPRESS (SQL Server 12.0.0)'. The Results pane displays the output of a SELECT query:

```
SELECT * FROM EMPLOYEE
```

| EMPLOYEE_ID | EMPLOYEE_NAME | EMAIL_ID | CONTACT_NO | DESIGNATION | DEPARTMENT | WORK_LOCATION | EXPENSE_LIMIT | APPROVAL_LIMIT |
|-------------|----------------|-----------------------|--------------|--------------|------------|---------------|---------------|----------------|
| 1000 | John Henks | jhenks@riotgames.com | 406-872-2919 | Associate | Marketing | Santa Clara | 5000.00 | NULL |
| 1001 | Meera M | meeram@riotgames.com | 406-467-2810 | Manager | Technology | Mipitas | 15000.00 | 20000.00 |
| 1002 | Rohit P | rohitp@riotgames.com | 916-693-2811 | Associate | Sales | New York | 5000.00 | NULL |
| 1003 | Frank Mank | frankm@riotgames.com | 607-727-6597 | Manager | Technology | Santa Clara | 15000.00 | 20000.00 |
| 1004 | Jade Turel | jodel@riotgames.com | 406-467-2813 | Associate | Sales | Mipitas | 5000.00 | NULL |
| 1005 | Venita Pandey | venitap@riotgames.com | 406-467-2814 | Associate | Technology | Mipitas | 5000.00 | NULL |
| 1006 | Rishabh Apte | rishabh@riotgames.com | 607-565-6685 | Associate | Accounting | New York | 5000.00 | NULL |
| 1007 | Robert McCarra | robertm@riotgames.com | 406-467-2816 | Associate | Technology | New York | 5000.00 | NULL |
| 1008 | Ashly John | ashly@riotgames.com | 607-727-6595 | Manager | Accounting | San Francisco | 15000.00 | 20000.00 |
| 1009 | Jon Hansen | jonh@riotgames.com | 916-467-2913 | Associate | Production | Mipitas | 5000.00 | NULL |
| 1010 | Nelly Gelmon | nelyg@riotgames.com | 916-364-2915 | Sr Analyst | Technology | New York | 10000.00 | NULL |
| 1011 | Rohit Bhatia | rohitbh@riotgames.com | 406-467-2820 | Manager | Technology | Mipitas | 15000.00 | 20000.00 |
| 1012 | Lisa Bell | lisab@riotgames.com | 916-467-2821 | Associate | Sales | Mipitas | 5000.00 | NULL |
| 1013 | Alice Jouston | alco@riotgames.com | 406-881-4983 | Associate | Technology | San Francisco | 5000.00 | NULL |
| 1014 | Aah Grover | aahg@riotgames.com | 916-767-2888 | Sr Analyst | Sales | Santa Clara | 10000.00 | NULL |
| 1015 | Neha Goval | nehag@riotgames.com | 406-411-2210 | Associate | Technology | New York | 5000.00 | NULL |
| 1016 | Taylor G | tayforg@riotgames.com | 607-761-1132 | Sr Associate | Accounting | San Diego | 10000.00 | NULL |
| 1017 | Simon K | simonk@riotgames.com | 916-364-2123 | Analyst | Admin | San Francisco | 10000.00 | NULL |
| 1018 | Darren Graph | darelg@riotgames.com | 607-737-2585 | Associate | Marketing | San Diego | 5000.00 | NULL |
| 1019 | Rinky K | rinkyk@riotgames.com | 406-466-2800 | Manager | Marketing | San Diego | 15000.00 | 20000.00 |
| 1020 | Lossie Pace | lossiep@riotgames.com | 916-365-2124 | Associate | Admin | San Francisco | 5000.00 | NULL |
| 1021 | Bella Swan | bella@riotgames.com | 406-678-900 | Sr Manager | Technology | Mipitas | 30000.00 | 50000.00 |
| 1022 | Edward Cullen | edwardc@riotgames.com | 775-896-234 | Sr Manager | Accounting | San Francisco | 30000.00 | 50000.00 |
| 1023 | Alice Grahams | aliceg@riotgames.com | 916-101-2345 | Sr Manager | Technology | Mipitas | 30000.00 | 50000.00 |
| 1024 | Xiu Zhang | xiaz@riotgames.com | 607-889-2456 | Manager | Production | Mipitas | 15000.00 | 20000.00 |
| 1025 | Akshay S | akshays@riotgames.com | 510-245-9876 | Manager | Sales | Santa Clara | 15000.00 | 20000.00 |
| 1026 | Ping Zhe | pingz@riotgames.com | 510-454-8080 | Manager | Admin | San Francisco | 15000.00 | 20000.00 |
| 1027 | Jack Nicholas | jackn@riotgames.com | 406-929-3458 | Associate | Accounting | San Francisco | 5000.00 | NULL |
| 1028 | Jimmy Jade | jimmyj@riotgames.com | 406-206-9198 | Associate | Accounting | New York | 5000.00 | NULL |

Query executed successfully.

QUERY PROCESSING

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer pane shows a connection to 'RISHABH\SOLEXPRESS (SQL Server 12.0.0)'. The Results pane displays the output of a DELETE query:

```
DELETE FROM EMPLOYEE WHERE Employee_ID = '1028'
```

(1 row(s) affected)

Query executed successfully.

AFTER PROCESSING THE QUERY

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery17.sql - RISHABH\SQLEXPRESS.master (RISHABH\Rishabh (53)) - Microsoft SQL Server Management Studio". The main window displays the results of a query:

```

SELECT
FROM EMPLOYEE

```

The results grid shows 28 rows of employee data. The columns are:

| | EMPLOYEE_ID | EMPLOYEE_NAME | EMAIL_ID | CONTACT_NO | DESIGNATION | DEPARTMENT | WORK_LOCATION | EXPENSE_LIMIT | APPROVAL_LIMIT |
|----|-------------|----------------|------------------------|--------------|--------------|------------|---------------|---------------|----------------|
| 1 | 1000 | John Henk | jhenk@riotgames.com | 408-572-2919 | Associate | Marketing | Santa Clara | 5000.00 | NULL |
| 2 | 1001 | Meena M | mearam@riotgames.com | 408-467-2810 | Manager | Technology | Milpitas | 15000.00 | 20000.00 |
| 3 | 1002 | Rohit P | rohitp@riotgames.com | 916-693-2811 | Associate | Sales | New York | 5000.00 | NULL |
| 4 | 1003 | Frank Mank | frankm@riotgames.com | 607-727-6597 | Manager | Technology | Santa Clara | 15000.00 | 20000.00 |
| 5 | 1004 | Jade Turel | jade@riotgames.com | 408-467-2813 | Associate | Sales | Milpitas | 5000.00 | NULL |
| 6 | 1005 | Venita Pandey | venitap@riotgames.com | 408-467-2814 | Associate | Technology | Milpitas | 5000.00 | NULL |
| 7 | 1006 | Rishabh Apte | rishabh@riotgames.com | 607-565-6685 | Associate | Accounting | New York | 5000.00 | NULL |
| 8 | 1007 | Robert McCorin | robertmc@riotgames.com | 408-467-2818 | Associate | Technology | New York | 5000.00 | NULL |
| 9 | 1008 | Ashly John | ashlyj@riotgames.com | 607-727-6595 | Manager | Accounting | San Francisco | 15000.00 | 20000.00 |
| 10 | 1009 | Jon Hansen | jonhy@riotgames.com | 916-467-2913 | Associate | Production | Milpitas | 5000.00 | NULL |
| 11 | 1010 | Nelly Gofman | nellyg@riotgames.com | 916-364-2915 | Sr Analyst | Technology | New York | 10000.00 | NULL |
| 12 | 1011 | Rohit Bhatia | rohitbh@riotgames.com | 408-467-2820 | Manager | Technology | Milpitas | 15000.00 | 20000.00 |
| 13 | 1012 | Lise Boltz | lisab@riotgames.com | 916-467-2821 | Associate | Sales | Milpitas | 5000.00 | NULL |
| 14 | 1013 | Alice Joshton | alicej@riotgames.com | 408-981-4983 | Associate | Technology | San Francisco | 5000.00 | NULL |
| 15 | 1014 | Ash Grover | ashg@riotgames.com | 916-767-2888 | Sr Analyst | Sales | Santa Clara | 10000.00 | NULL |
| 16 | 1015 | Neha Goyle | nehag@riotgames.com | 408-411-2210 | Associate | Technology | New York | 5000.00 | NULL |
| 17 | 1016 | Taylor G | tayforg@riotgames.com | 607-761-1132 | Sr Associate | Accounting | San Diego | 10000.00 | NULL |
| 18 | 1017 | Simon K | simonk@riotgames.com | 916-364-2123 | Analyst | Admin | San Francisco | 10000.00 | NULL |
| 19 | 1018 | Darrel Graph | darrelg@riotgames.com | 607-737-2585 | Associate | Marketing | San Diego | 5000.00 | NULL |
| 20 | 1019 | Rinky K | rinkyk@riotgames.com | 408-466-2800 | Manager | Marketing | San Diego | 15000.00 | 20000.00 |
| 21 | 1020 | Lossie Pace | lossiep@riotgames.com | 916-365-2124 | Associate | Admin | San Francisco | 5000.00 | NULL |
| 22 | 1021 | Belle Sween | belle@riotgames.com | 408-678-900 | Sr Manager | Technology | Milpitas | 30000.00 | 50000.00 |
| 23 | 1022 | Edward Cullen | edwardc@riotgames.com | 775-986-234 | Sr Manager | Accounting | San Francisco | 30000.00 | 50000.00 |
| 24 | 1023 | Alice Graham | aliceg@riotgames.com | 916-101-2345 | Sr Manager | Technology | Milpitas | 30000.00 | 50000.00 |
| 25 | 1024 | Xiu Zhang | xuz@riotgames.com | 607-889-2456 | Manager | Production | Milpitas | 15000.00 | 20000.00 |
| 26 | 1025 | Akshay S | akshays@riotgames.com | 510-245-9876 | Manager | Sales | Santa Clara | 15000.00 | 20000.00 |
| 27 | 1026 | Ping Zha | pingz@riotgames.com | 510-454-8080 | Manager | Admin | San Francisco | 15000.00 | 20000.00 |
| 28 | 1027 | Jack Nicholas | jackn@riotgames.com | 408-929-3456 | Associate | Accounting | San Francisco | 5000.00 | NULL |

Below the results grid, a message bar says "Query executed successfully." The status bar at the bottom right shows "RISHABH\SQLEXPRESS (12.0 RTM) RISHABH\Rishabh (53) master 00:00:00 28 rows".

Business Process 2:

When there is a requirement for items like PCs, laptops, routers, etc. an employee submits a purchase request which gets directed to his manager. The manager reviews the request and on approval of the request, an associate from the Procurement and Payment associate team picks it up. He then creates a purchase order and the order gets assigned to a supplier.

Step 1: CREATE Purchase Request for EMPLOYEE_ID 1027

BEFORE EXECUTING THE QUERY

The screenshot shows a SQL Server Management Studio window with four tabs at the top: 'SQLQuery7.sql - RI...HABH\Rishabh (59)*', 'script.sql - RISHA...SHABH\Rishabh (55)', 'SQLQuery4.sql - RI...HABH\Rishabh (54)*', and 'SQLQuery2.sql - RI...HABH\Rishabh (53)*'. The 'Results' tab is selected. A query is being run:

```
SELECT*
FROM PURCHASE_REQUEST
```

The results grid displays 36 rows of data from the PURCHASE_REQUEST table. The columns are: PURCHASE_REQUEST_ID, REQUESTOR_EMPLOYEE_ID, APPROVED_EMPLOYEE_ID, REQUEST_STATUS, ASSIGNED_EMPLOYEE_ID, REQUEST_SUBMITTED_DATE, TOTAL_REQUEST_AMOUNT, APPROVER_REVIEW_DATE, DEPARTMENT, and WORK_LOCATION. The data includes various employee IDs, dates ranging from 2015-01-18 to 2015-09-18, and department names like Accounting, Technology, Sales, Admin, Marketing, and Santa Clara.

At the bottom of the results grid, a message says: "Query executed successfully."

QUERY PROCESSING

```

SQLQuery12.sql -...HABH\Rishabh (57)* SQLQuery10.sql -...HABH\Rishabh (56)* SQLQuery9.sql - RI...HABH\Rishabh (59)* scripts.sql - RISHABH\Rishabh (55)* SQLQuery4.sql - RI...HABH\Rishabh (54)*
INSERT INTO PURCHASE_REQUEST(PURCHASE_REQUEST_ID, REQUESTOR_ID, REQUEST_STATUS, ASSIGNED_TO, REQUEST_SUBMITTED_DATE, TOTAL_REQUEST_AMOUNT, DEPARTMENT, WORK_LOCATION)
VALUES ('437', '1027', 'Open', '1016', '05/04/2016', '1340', 'Accounting', 'San Francisco')

(1 row(s) affected)

```

AFTER THE QUERY IS PROCESSED

```

SQLQuery16.sql -...HABH\Rishabh (60)* SQLQuery15.sql -...HABH\Rishabh (58)* SQLQuery12.sql -...HABH\Rishabh (57)* SQLQuery10.sql -...HABH\Rishabh (56)* SQLQuery9.sql - RI...HABH\Rishabh (59)*
SELECT*
FROM PURCHASE_REQUEST

INSERT INTO PURCHASE_REQUEST(PURCHASE_REQUEST_ID, REQUESTOR_ID, REQUEST_STATUS, ASSIGNED_TO, REQUEST_SUBMITTED_DATE, TOTAL_REQUEST_AMOUNT, DEPARTMENT, WORK_LOCATION)
VALUES ('437', '1027', 'Open', '1016', '05/04/2016', '1340', 'Accounting', 'San Francisco')

```

| PURCHASE_REQUEST_ID | REQUESTOR_ID | APPROVED_BY | REQUEST_STATUS | ASSIGNED_TO | REQUEST_SUBMITTED_DATE | TOTAL_REQUEST_AMOUNT | APPROVER_REVIEW_DATE | DEPARTMENT | WORK_LOCATION | |
|---------------------|--------------|-------------|----------------|-------------|------------------------|-------------------------|----------------------|-------------------------|---------------|---------------|
| 11 | 411 | 1010 | 1011 | Approved | 1006 | 2016-04-27 00:00:00.000 | 3500.00 | 2016-04-27 00:00:00.000 | Technology | New York |
| 12 | 412 | 1011 | 1021 | Rejected | 1006 | 2016-01-28 00:00:00.000 | 11000.00 | 2016-01-29 00:00:00.000 | Technology | Milpitas |
| 13 | 413 | 1012 | 1025 | Open | 1006 | 2015-09-22 00:00:00.000 | 4000.00 | NULL | Sales | Milpitas |
| 14 | 414 | 1013 | 1008 | Rejected | 1006 | 2015-08-27 00:00:00.000 | 1400.00 | 2015-08-29 00:00:00.000 | Technology | San Francisco |
| 15 | 415 | 1014 | 1025 | Approved | 1016 | 2015-05-30 00:00:00.000 | 3400.00 | 2015-06-02 00:00:00.000 | Sales | Santa Clara |
| 16 | 416 | 1015 | 1011 | Rejected | 1008 | 2015-11-28 00:00:00.000 | 3000.00 | 2015-11-28 00:00:00.000 | Technology | New York |
| 17 | 417 | 1016 | 1008 | Open | 1016 | 2015-05-24 00:00:00.000 | 5500.00 | NULL | Accounting | San Diego |
| 18 | 418 | 1017 | 1026 | Open | 1006 | 2015-10-25 00:00:00.000 | 7000.00 | NULL | Admin | San Francisco |
| 19 | 419 | 1018 | 1019 | Approved | 1006 | 2015-12-12 00:00:00.000 | 250.00 | 2015-12-12 00:00:00.000 | Marketing | San Diego |
| 20 | 420 | 1019 | NULL | Approved | 1008 | 2015-12-18 00:00:00.000 | 800.00 | 1900-01-01 00:00:00.000 | Marketing | San Diego |
| 21 | 421 | 1020 | 1026 | Rejected | 1008 | 2016-05-09 00:00:00.000 | 3000.00 | 2016-05-10 00:00:00.000 | Admin | San Francisco |
| 22 | 422 | 1001 | 1021 | Approved | 1016 | 2016-05-12 00:00:00.000 | 5000.00 | 2016-05-13 00:00:00.000 | Technology | Milpitas |
| 23 | 423 | 1004 | 1025 | Approved | 1008 | 2016-04-10 00:00:00.000 | 2000.00 | 2016-04-10 00:00:00.000 | Sales | Milpitas |
| 24 | 424 | 1008 | 1022 | Approved | 1006 | 2016-03-08 00:00:00.000 | 400.00 | 2016-03-10 00:00:00.000 | Accounting | San Francisco |
| 25 | 425 | 1012 | 1025 | Approved | 1016 | 2015-12-28 00:00:00.000 | 1100.00 | 2015-12-30 00:00:00.000 | Sales | Milpitas |
| 26 | 426 | 1003 | 1023 | Open | 1008 | 2015-11-09 00:00:00.000 | 3500.00 | 2015-11-11 00:00:00.000 | Technology | Santa Clara |
| 27 | 427 | 1009 | 1024 | Rejected | 1006 | 2015-10-25 00:00:00.000 | 4200.00 | 2015-10-28 00:00:00.000 | Production | Milpitas |
| 28 | 428 | 1012 | 1025 | Approved | 1016 | 2016-01-18 00:00:00.000 | 2100.00 | 2016-01-19 00:00:00.000 | Sales | Milpitas |
| 29 | 429 | 1005 | 1003 | Open | 1008 | 2016-04-23 00:00:00.000 | 4400.00 | 2016-04-23 00:00:00.000 | Technology | Milpitas |
| 30 | 430 | 1019 | NULL | Approved | 1016 | 2015-09-18 00:00:00.000 | 4900.00 | 2015-09-21 00:00:00.000 | Marketing | San Diego |
| 31 | 431 | 1014 | 1025 | Open | 1008 | 2015-08-12 00:00:00.000 | 1800.00 | 2015-08-14 00:00:00.000 | Sales | Santa Clara |
| 32 | 432 | 1017 | 1026 | Approved | 1006 | 2016-02-10 00:00:00.000 | 8000.00 | 2016-02-10 00:00:00.000 | Admin | San Francisco |
| 33 | 433 | 1002 | 1025 | Rejected | 1016 | 2015-07-07 00:00:00.000 | 1450.00 | 2015-07-09 00:00:00.000 | Sales | New York |
| 34 | 434 | 1011 | 1021 | Approved | 1006 | 2015-06-19 00:00:00.000 | 6000.00 | 2015-06-23 00:00:00.000 | Technology | Milpitas |
| 35 | 435 | 1008 | 1008 | Approved | 1006 | 2015-05-24 00:00:00.000 | 3300.00 | 2015-05-25 00:00:00.000 | Accounting | New York |
| 36 | 436 | 1027 | 1008 | Closed | 1008 | 2016-04-01 00:00:00.000 | 2275.00 | 2016-04-04 00:00:00.000 | Accounting | San Francisco |
| 37 | 437 | 1027 | NULL | Open | 1016 | 2016-05-04 00:00:00.000 | 1340.00 | NULL | Accounting | San Francisco |

Query executed successfully.

Step 2: Add PURCHASE_REQUEST_LINE_ITEMS for PURCHASE_REQUEST_ID 437

```
SQLQuery18.sql -...HABH\Rishabh (62)* SQLQuery16.sql -...HABH\Rishabh (60)* SQLQuery15.sql -...HABH\Rishabh (58)* SQLQuery12.sql -...HABH\Rishabh (57)* SQLQuery10.sql -...HABH\Rishabh (55)*
100 % 
1 Messages
(1 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)
```

```
SQLQuery18.sql -...HABH\Rishabh (62)* SQLQuery16.sql -...HABH\Rishabh (60)* SQLQuery15.sql -...HABH\Rishabh (58)* SQLQuery12.sql -...HABH\Rishabh (57)* SQLQuery10.sql -...HABH\Rishabh (55)*
100 % 
1 Messages
VALUES ('437', '1', '305', '5', '500')
INSERT INTO PURCHASE_REQUEST_LINE_ITEMS (PURCHASE_REQUEST_ID, PURCHASE_REQUEST_LINE_ITEM_ID, ITEM_CODE, ITEM_QUANTITY_ORDERED, TOTAL_PRICE)
VALUES ('437', '2', '302', '10', '10000')
VALUES ('437', '3', '319', '4', '4800')

SELECT *
FROM PURCHASE_REQUEST_LINE_ITEMS PR
WHERE PR.PURCHASE_REQUEST_ID = '437'
```

| PURCHASE_REQUEST | PURCHASE_REQUEST_LINE_ITEM_ID | ITEM_CODE | ITEM_QUANTITY_ORDERED | TOTAL_PRICE |
|------------------|-------------------------------|-----------|-----------------------|-------------|
| 1 | 1 | 305 | 5 | 500.00 |
| 2 | 2 | 302 | 10 | 10000.00 |
| 3 | 3 | 319 | 4 | 4800.00 |

Step 3: Purchase request status changes to 'APPROVED' for PURCHASE_REQUEST_ID 437

QUERY PROCESSING

```
SQLQuery16.sql -...HABH\Rishabh (60)* SQLQuery15.sql -...HABH\Rishabh (58)* SQLQuery12.sql -...HABH\Rishabh (57)*
100 %
1 Messages
(1 row(s) affected)
```

AFTER THE QUERY IS PROCESSED

The screenshot shows a SQL Server Management Studio window with multiple tabs at the top. The active tab is 'SQLQuery15.sql - HABH\Rishabh (58)*'. The query in the editor is:

```
SELECT *
FROM PURCHASE_REQUEST PR
WHERE PR.PURCHASE_REQUEST_ID = '437'
```

The results pane shows one row of data:

| PURCHASE_REQUEST_ID | REQUESTOR_ID | APPROVED_BY | REQUEST_STATUS | ASSIGNED_TO | REQUEST_SUBMITTED_DATE | TOTAL_REQUEST_AMOUNT | APPROVER_REVIEW_DATE | DEPARTMENT | WORK_LOCATION |
|---------------------|--------------|-------------|----------------|-------------|-------------------------|----------------------|-------------------------|------------|---------------|
| 437 | 1027 | 1008 | Approved | 1016 | 2016-05-04 00:00:00.000 | 1340.00 | 2016-05-05 00:00:00.000 | Accounting | San Francisco |

Step 4: An associate from purchase and procurement team creates a purchase order

CREATE PURCHASE ORDER

ISIS-2603-Project-Expensify (RISHABH\Rishabh (60)* - Microsoft SQL Server Management Studio

The screenshot shows a SQL Server Management Studio window with multiple tabs at the top. The active tab is 'SQLQuery16.sql - HABH\Rishabh (60)*'. The query in the editor is:

```
INSERT INTO PURCHASE_ORDER(ORDER_ID, PURCHASE_REQUEST_ID, ORDER_CREATED_DATE, CREATED_BY, ASSIGNED_TO, DEPARTMENT, WORK_LOCATION, ORDER_STATUS)
VALUES ('522', '437', '05/07/2016', '1015', '201', 'Accounting', 'San Francisco', 'Open')
```

The results pane shows the message '(1 row(s) affected)'.

AFTER QUERY IS PROCESSED

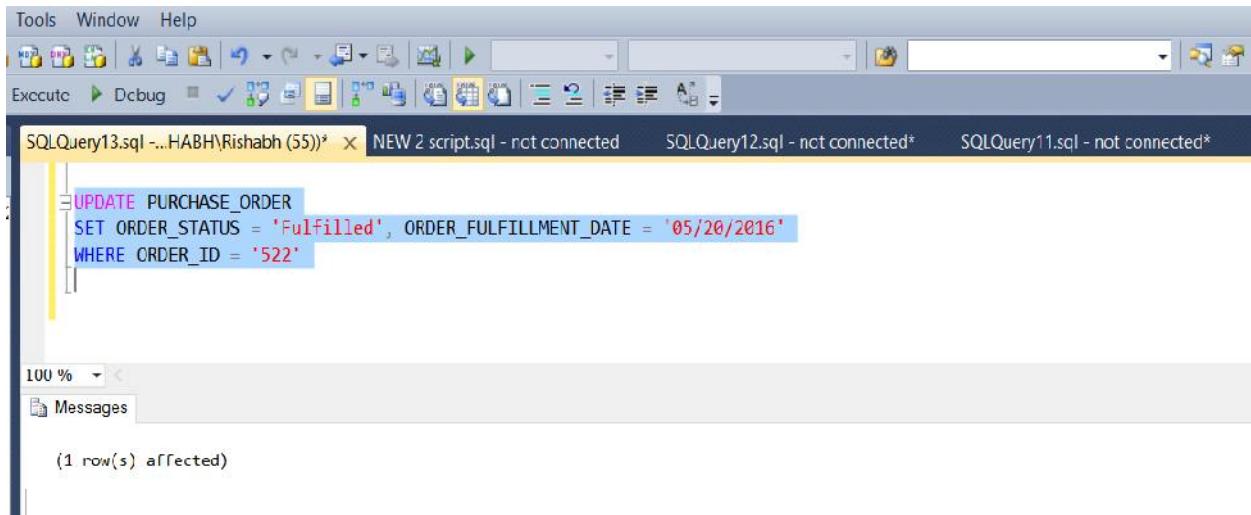
The screenshot shows a SQL Server Management Studio window with multiple tabs at the top. The active tab is 'SQLQuery18.sql - HABH\Rishabh (52)*'. The query in the editor is:

```
SELECT *
FROM PURCHASE_ORDER PO
WHERE PO.ORDER_ID = '522'
```

The results pane shows one row of data:

| ORDER_ID | PURCHASE_REQUEST_ID | ORDER_CREATED_DATE | ORDER_FULFILLMENT_DATE | CREATED_BY | ASSIGNED_TO | DEPARTMENT | WORK_LOCATION | ORDER_STATUS | ORDER_COMPLETED_DATE |
|----------|---------------------|-------------------------|------------------------|------------|-------------|------------|---------------|--------------|----------------------|
| 522 | 437 | 2016-05-07 00:00:00.000 | NULL | 1015 | 201 | Accounting | San Francisco | Open | NULL |

Step 5: Supplier updates the status periodically once he starts working on the order. Once the supplier fulfills the order, he updates the order status as well as sends an invoice to the purchase and procurement team

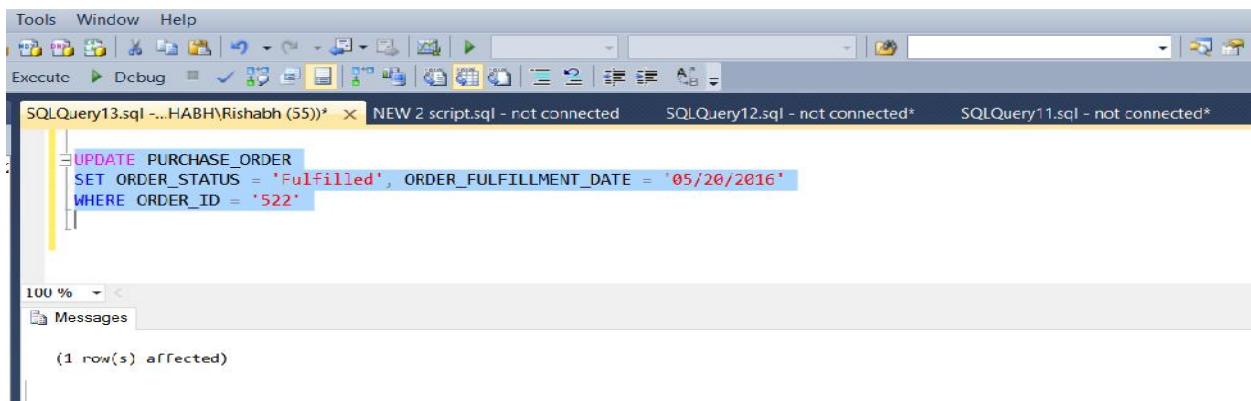


The screenshot shows the SQL Server Management Studio interface. The toolbar at the top includes icons for Tools, Window, Help, and various database management functions. Below the toolbar, the title bar displays multiple open connections: 'SQLQuery13.sql -... HABH\Rishabh (55)*' (highlighted in yellow), 'NEW 2 script.sql - not connected', 'SQLQuery12.sql - not connected*', and 'SQLQuery11.sql - not connected*'. The main query window contains the following SQL code:

```
UPDATE PURCHASE_ORDER
SET ORDER_STATUS = 'Fulfilled', ORDER_FULFILLMENT_DATE = '05/20/2016'
WHERE ORDER_ID = '522'
```

Below the query window, the status bar shows '100 %' and a 'Messages' button. The message pane below the status bar displays the result: '(1 row(s) affected)'.

AFTER PROCESSING THE QUERY

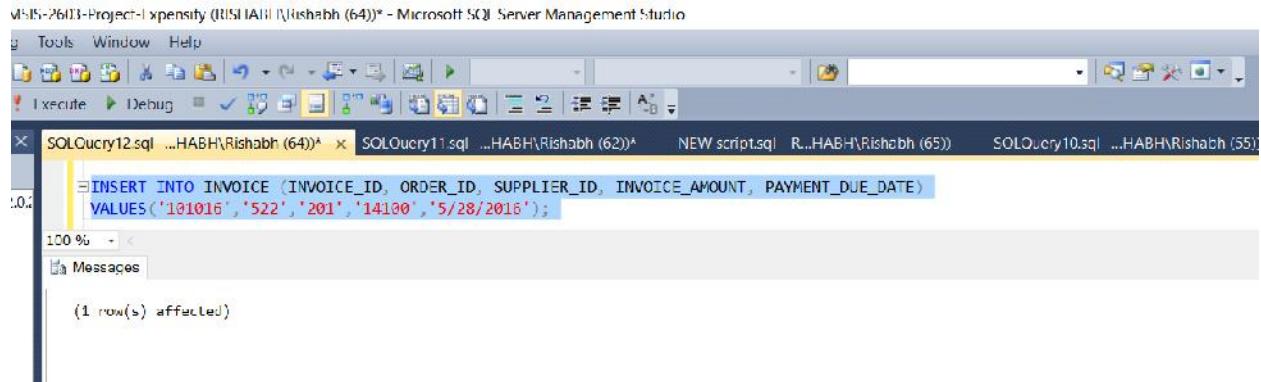


This screenshot is identical to the previous one, showing the same SQL Server Management Studio interface and the same UPDATE query. The status bar still shows '100 %' and the 'Messages' button. The message pane also still displays '(1 row(s) affected)'.

Step 6: On delivery of the order, purchase and procurement associate verifies purchase order details with invoice details and creates an invoice record in system.

INVOICE CREATION

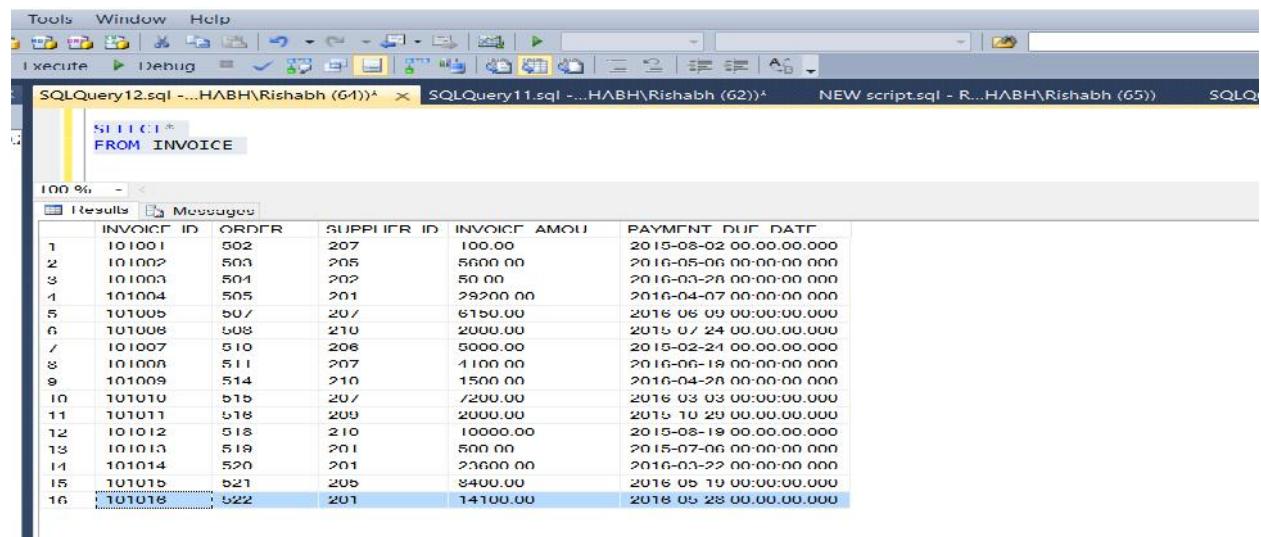
```
INSERT INTO INVOICE (INVOICE_ID, ORDER_ID, SUPPLIER_ID, INVOICE_AMOUNT, PAYMENT_DUE_DATE)
VALUES('101016', '522', '201', '14100', '5/28/2016');
```



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window is open with the following SQL code:

```
INSERT INTO INVOICE (INVOICE_ID, ORDER_ID, SUPPLIER_ID, INVOICE_AMOUNT, PAYMENT_DUE_DATE)
VALUES('101016', '522', '201', '14100', '5/28/2016');
```

The status bar at the bottom indicates "100 %". Below the query window, the "Messages" pane shows the result: "(1 row(s) affected)".



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window is open with the following SQL code:

```
SELECT *
FROM INVOICE
```

The results pane displays a table of data from the INVOICE table:

| | INVOICE_ID | ORDER_ID | SUPPLIER_ID | INVOICE_AMOU | PAYMENT_DUE_DATE |
|----|------------|----------|-------------|--------------|-------------------------|
| 1 | 101001 | 502 | 207 | 100.00 | 2015-06-02 00:00:00.000 |
| 2 | 101002 | 503 | 205 | 5000.00 | 2016-05-06 00:00:00.000 |
| 3 | 101003 | 504 | 202 | 50.00 | 2016-03-28 00:00:00.000 |
| 4 | 101004 | 505 | 201 | 29200.00 | 2016-04-07 00:00:00.000 |
| 5 | 101005 | 507 | 207 | 6150.00 | 2016-06-09 00:00:00.000 |
| 6 | 101006 | 508 | 210 | 2000.00 | 2015-07-24 00:00:00.000 |
| 7 | 101007 | 510 | 206 | 5000.00 | 2015-02-24 00:00:00.000 |
| 8 | 101008 | 511 | 207 | 4100.00 | 2016-06-18 00:00:00.000 |
| 9 | 101009 | 514 | 210 | 1500.00 | 2016-04-26 00:00:00.000 |
| 10 | 101010 | 515 | 207 | 7200.00 | 2016-03-03 00:00:00.000 |
| 11 | 101011 | 516 | 209 | 2000.00 | 2015-10-29 00:00:00.000 |
| 12 | 101012 | 518 | 210 | 10000.00 | 2015-08-18 00:00:00.000 |
| 13 | 101013 | 519 | 201 | 500.00 | 2015-07-06 00:00:00.000 |
| 14 | 101014 | 520 | 201 | 20000.00 | 2016-03-22 00:00:00.000 |
| 15 | 101015 | 521 | 205 | 8400.00 | 2016-05-19 00:00:00.000 |
| 16 | 101016 | 522 | 201 | 14100.00 | 2016-05-28 00:00:00.000 |

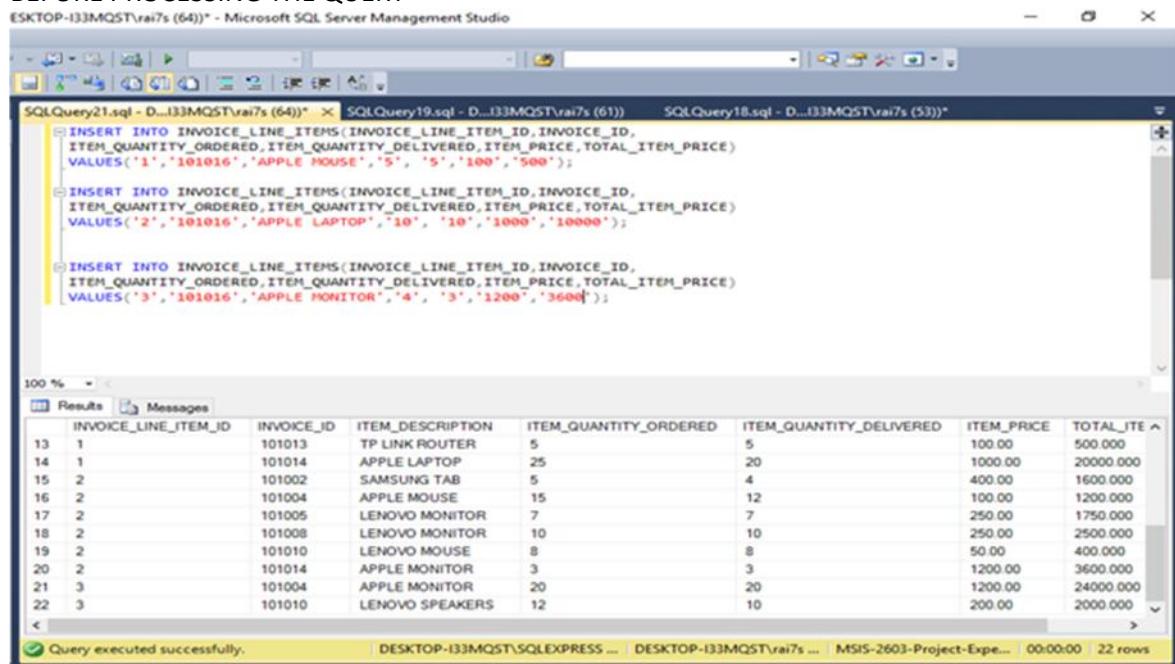
INSERT INVOICE LINE ITEMS

```
INSERT INTO INVOICE_LINE_ITEMS(INVOICE_LINE_ITEM_ID,INVOICE_ID,ITEM_DESCRIPTION,
ITEM_QUANTITY_ORDERED,ITEM_QUANTITY_DELIVERED,ITEM_PRICE,TOTAL_ITEM_PRICE)
VALUES('1','101016','APPLE MOUSE','5','5','100','500');
```

```
INSERT INTO INVOICE_LINE_ITEMS(INVOICE_LINE_ITEM_ID,INVOICE_ID,ITEM_DESCRIPTION,
ITEM_QUANTITY_ORDERED,ITEM_QUANTITY_DELIVERED,ITEM_PRICE,TOTAL_ITEM_PRICE)
VALUES('2','101016','APPLE LAPTOP','10','10','1000','10000');
```

```
INSERT INTO INVOICE_LINE_ITEMS(INVOICE_LINE_ITEM_ID,INVOICE_ID,ITEM_DESCRIPTION,
ITEM_QUANTITY_ORDERED,ITEM_QUANTITY_DELIVERED,ITEM_PRICE,TOTAL_ITEM_PRICE)
VALUES('3','101016','APPLE MONITOR','4','3','1200','3600');
```

BEFORE PROCESSING THE QUERY



The screenshot shows the Microsoft SQL Server Management Studio interface with three tabs open in the background:

- SQLQuery21.sql - D...I33MQST\rai7s (64))*
- SQLQuery19.sql - D...I33MQST\rai7s (61))
- SQLQuery18.sql - D...I33MQST\rai7s (53))*

The foreground window displays three separate `INSERT INTO` statements for the `INVOICE_LINE_ITEMS` table:

```
INSERT INTO INVOICE_LINE_ITEMS(INVOICE_LINE_ITEM_ID,INVOICE_ID,ITEM_DESCRIPTION,
ITEM_QUANTITY_ORDERED,ITEM_QUANTITY_DELIVERED,ITEM_PRICE,TOTAL_ITEM_PRICE)
VALUES('1','101016','APPLE MOUSE','5','5','100','500');

INSERT INTO INVOICE_LINE_ITEMS(INVOICE_LINE_ITEM_ID,INVOICE_ID,ITEM_DESCRIPTION,
ITEM_QUANTITY_ORDERED,ITEM_QUANTITY_DELIVERED,ITEM_PRICE,TOTAL_ITEM_PRICE)
VALUES('2','101016','APPLE LAPTOP','10','10','1000','10000');

INSERT INTO INVOICE_LINE_ITEMS(INVOICE_LINE_ITEM_ID,INVOICE_ID,ITEM_DESCRIPTION,
ITEM_QUANTITY_ORDERED,ITEM_QUANTITY_DELIVERED,ITEM_PRICE,TOTAL_ITEM_PRICE)
VALUES('3','101016','APPLE MONITOR','4','3','1200','3600');
```

Below the code, the results pane shows a table with 22 rows of data from another query. The columns are:

| | INVOICE_LINE_ITEM_ID | INVOICE_ID | ITEM_DESCRIPTION | ITEM_QUANTITY_ORDERED | ITEM_QUANTITY_DELIVERED | ITEM_PRICE | TOTAL_ITEM_PRICE |
|----|----------------------|------------|------------------|-----------------------|-------------------------|------------|------------------|
| 13 | 1 | 101013 | TP LINK ROUTER | 5 | 5 | 100.00 | 500.00 |
| 14 | 1 | 101014 | APPLE LAPTOP | 25 | 20 | 1000.00 | 20000.00 |
| 15 | 2 | 101002 | SAMSUNG TAB | 5 | 4 | 400.00 | 1600.00 |
| 16 | 2 | 101004 | APPLE MOUSE | 15 | 12 | 100.00 | 1200.00 |
| 17 | 2 | 101005 | LENOVO MONITOR | 7 | 7 | 250.00 | 1750.00 |
| 18 | 2 | 101008 | LENOVO MONITOR | 10 | 10 | 250.00 | 2500.00 |
| 19 | 2 | 101010 | LENOVO MOUSE | 8 | 8 | 50.00 | 400.00 |
| 20 | 2 | 101014 | APPLE MONITOR | 3 | 3 | 1200.00 | 3600.00 |
| 21 | 3 | 101004 | APPLE MONITOR | 20 | 20 | 1200.00 | 24000.00 |
| 22 | 3 | 101010 | LENOVO SPEAKERS | 12 | 10 | 200.00 | 2000.00 |

At the bottom of the results pane, it says "Query executed successfully." and shows the execution time as 00:00:00.

QUERY PROCESSING

```

SQLQuery21.sql - D...I33MQ5T\rai7s (64)* × SQLQuery19.sql - D...I33MQ5T\rai7s (61))      SQLQuery18.sql - D...I33MQ5T\rai7s (53))
INSERT INTO INVOICE_LINE_ITEMS(INVOICE_LINE_ITEM_ID,INVOICE_ID,ITEM_DESCRIPTION,
ITEM_QUANTITY_ORDERED,ITEM_QUANTITY_DELIVERED,ITEM_PRICE,TOTAL_ITEM_PRICE)
VALUES('1','101016','APPLE MOUSE','5', '5','100','500');

INSERT INTO INVOICE_LINE_ITEMS(INVOICE_LINE_ITEM_ID,INVOICE_ID,ITEM_DESCRIPTION,
ITEM_QUANTITY_ORDERED,ITEM_QUANTITY_DELIVERED,ITEM_PRICE,TOTAL_ITEM_PRICE)
VALUES('2','101016','APPLE LAPTOP','10', '10','1000','10000');

INSERT INTO INVOICE_LINE_ITEMS(INVOICE_LINE_ITEM_ID,INVOICE_ID,ITEM_DESCRIPTION,
ITEM_QUANTITY_ORDERED,ITEM_QUANTITY_DELIVERED,ITEM_PRICE,TOTAL_ITEM_PRICE)
VALUES('3','101016','APPLE MONITOR','4', '3','1200','3600');

```

Messages

```

(1 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)

```

```

SELECT*
FROM INVOICE_LINE_ITEMS

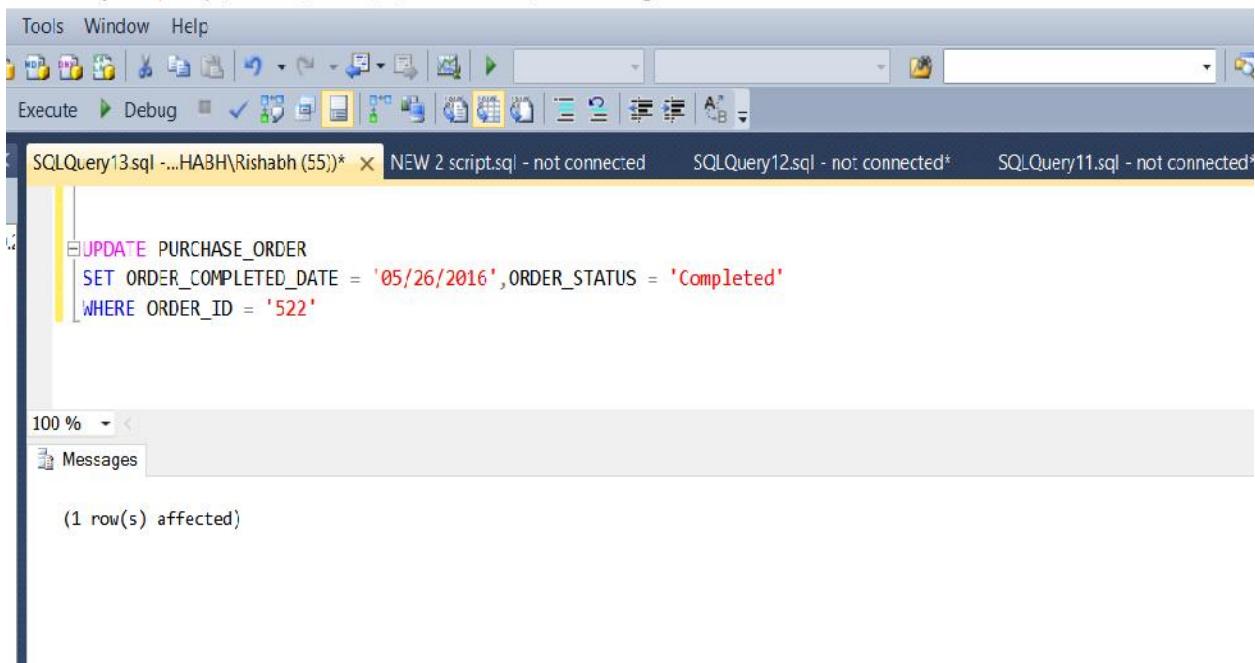
```

Results

| | INVOICE_LINE_ITEM_ID | INVOICE_ID | ITEM_DESCRIPTION | ITEM_QUANTITY_ORDERED | ITEM_QUANTITY_DELIVERED | ITEM_PRICE | TOTAL_ITEM_PRICE |
|----|----------------------|------------|-------------------|-----------------------|-------------------------|------------|------------------|
| 1 | 1 | 101001 | LENOVO MOUSE | 2 | 2 | 50.00 | 100.000 |
| 2 | 1 | 101002 | SAMSUNG WEBCAM | 5 | 5 | 800.00 | 4000.000 |
| 3 | 1 | 101003 | LOGITECH KEYBOARD | 1 | 1 | 50.00 | 50.000 |
| 4 | 1 | 101004 | APPLE LAPTOP | 4 | 4 | 1000.00 | 4000.000 |
| 5 | 1 | 101005 | LENOVO SPEAKERS | 25 | 22 | 200.00 | 4400.000 |
| 6 | 1 | 101006 | TP LINK ROUTER | 20 | 20 | 100.00 | 2000.000 |
| 7 | 1 | 101007 | SONY LAPTOP | 5 | 5 | 1000.00 | 5000.000 |
| 8 | 1 | 101008 | LENOVO SPEAKERS | 8 | 8 | 200.00 | 1600.000 |
| 9 | 1 | 101009 | TP LINK ROUTER | 15 | 15 | 100.00 | 1500.000 |
| 10 | 1 | 101010 | LENOVO LAPTOP | 6 | 6 | 800.00 | 4800.000 |
| 11 | 1 | 101011 | MOTOROLA MODEM | 20 | 20 | 100.00 | 2000.000 |
| 12 | 1 | 101012 | APPLE LAPTOP | 10 | 10 | 1000.00 | 10000.000 |
| 13 | 1 | 101013 | TP LINK ROUTER | 5 | 5 | 100.00 | 500.000 |
| 14 | 1 | 101014 | APPLE LAPTOP | 25 | 20 | 1000.00 | 20000.000 |
| 15 | 1 | 101015 | SAMSUNG TAB | 5 | 5 | 400.00 | 2000.000 |
| 16 | 1 | 101016 | APPLE MOUSE | 5 | 5 | 100.00 | 500.000 |
| 17 | 2 | 101002 | SAMSUNG TAB | 5 | 4 | 400.00 | 1600.000 |
| 18 | 2 | 101004 | APPLE MOUSE | 15 | 12 | 100.00 | 1200.000 |
| 19 | 2 | 101005 | LENOVO MONITOR | 7 | 7 | 250.00 | 1750.000 |
| 20 | 2 | 101008 | LENOVO MONITOR | 10 | 10 | 250.00 | 2500.000 |
| 21 | 2 | 101010 | LENOVO MOUSE | 8 | 8 | 50.00 | 400.000 |
| 22 | 2 | 101014 | APPLE MONITOR | 3 | 3 | 1200.00 | 3600.000 |
| 23 | 2 | 101015 | SAMSUNG WEBCAM | 10 | 8 | 800.00 | 6400.000 |
| 24 | 2 | 101016 | APPLE LAPTOP | 10 | 10 | 1000.00 | 10000.000 |
| 25 | 3 | 101004 | APPLE MONITOR | 20 | 20 | 1200.00 | 24000.000 |
| 26 | 3 | 101010 | LENOVO SPEAKERS | 12 | 10 | 200.00 | 2000.000 |
| 27 | 3 | 101016 | APPLE MONITOR | 4 | 3 | 1200.00 | 3600.000 |

Query executed successfully.

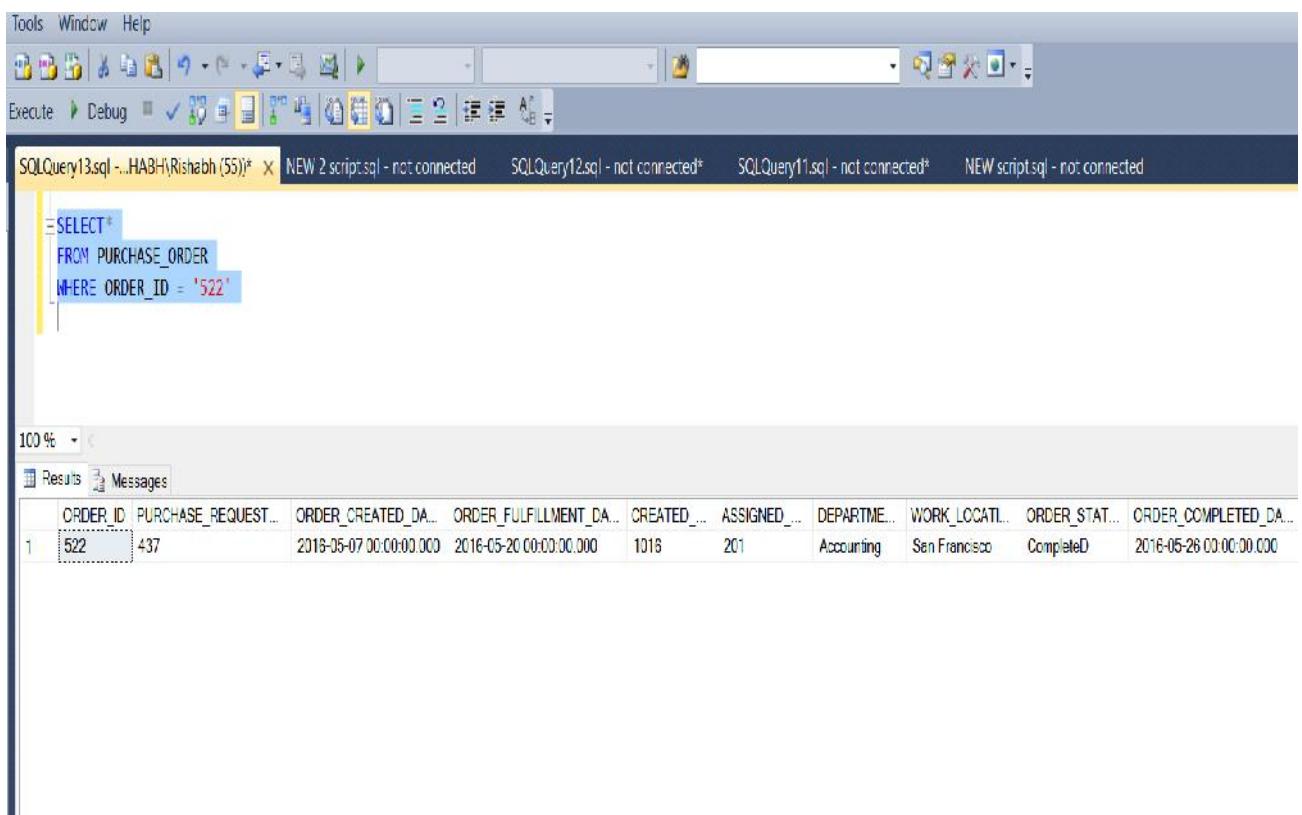
Step 7: After creating invoice, the Procurement and Payment associate marks the Purchase Order as completed and updates the Order completed date.



The screenshot shows the SQL Server Management Studio interface. In the center pane, there is a query window containing the following SQL code:

```
UPDATE PURCHASE_ORDER
SET ORDER_COMPLETED_DATE = '05/26/2016', ORDER_STATUS = 'Completed'
WHERE ORDER_ID = '522'
```

Below the query window, the status bar indicates "(1 row(s) affected)".



The screenshot shows the SQL Server Management Studio interface. In the center pane, there is a query window containing the following SQL code:

```
SELECT*
FROM PURCHASE_ORDER
WHERE ORDER_ID = '522'
```

Below the query window, the status bar indicates "Results". A results grid is displayed, showing the following data:

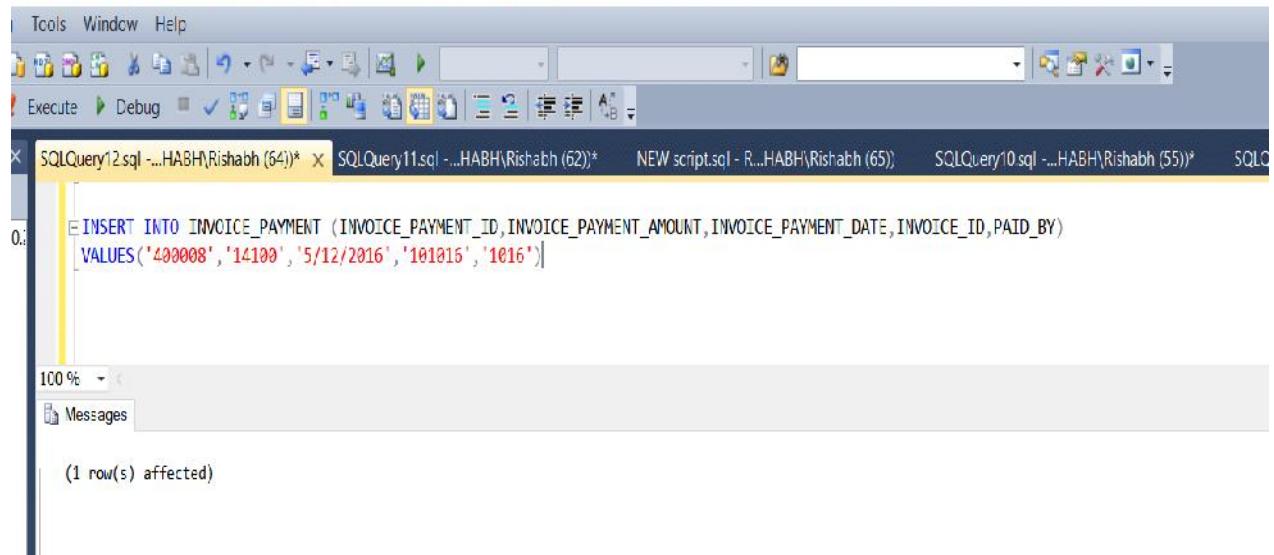
| | ORDER_ID | PURCHASE_REQUEST_ID | ORDER_CREATED_DATE | ORDER_FULFILLMENT_DATE | CREATED_BY | ASSIGNED_TO | DEPARTMENT | WORK_LOCATION | ORDER_STATUS | ORDER_COMPLETED_DATE |
|---|----------|---------------------|-------------------------|-------------------------|------------|-------------|------------|---------------|--------------|-------------------------|
| 1 | 522 | 437 | 2016-05-07 00:00:00.000 | 2016-05-20 00:00:00.000 | 1016 | 201 | Accounting | San Francisco | Completed | 2016-05-26 00:00:00.000 |

Step 8: On successful completion of the purchase order, Purchase and Procurement associate pays the supplier

Invoice payment

```
INSERT INTO INVOICE_PAYMENT  
(INVOICE_PAYMENT_ID,INVOICE_PAYMENT_AMOUNT,INVOICE_PAYMENT_DATE,INVOICE_ID,PAID_BY)  
VALUES('400008','14100','5/12/2016','101016','1016')
```

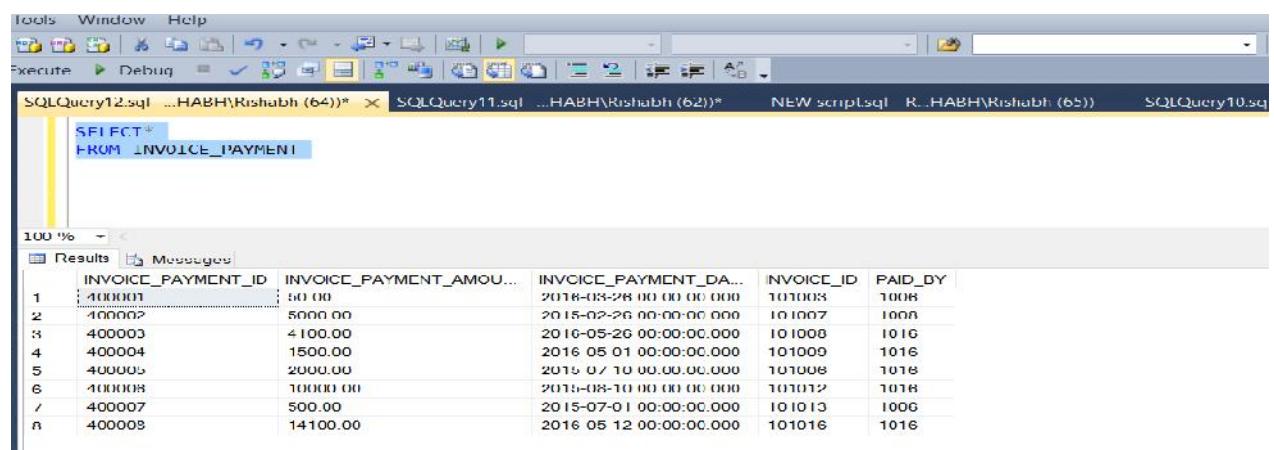
1SIS-2603-Project-Expensify [RISHABH\Rishabh (64)]* - Microsoft SQL Server Management Studio



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window is open with the following SQL code:

```
INSERT INTO INVOICE_PAYMENT (INVOICE_PAYMENT_ID,INVOICE_PAYMENT_AMOUNT,INVOICE_PAYMENT_DATE,INVOICE_ID,PAID_BY)  
VALUES('400008','14100','5/12/2016','101016','1016')
```

The status bar at the bottom indicates "(1 row(s) affected)".



The screenshot shows the Microsoft SQL Server Management Studio interface with a results grid. A query window is open with the following SQL code:

```
SELECT *  
FROM INVOICE_PAYMENT
```

The results grid displays the following data:

| | INVOICE_PAYMENT_ID | INVOICE_PAYMENT_AMOU... | INVOICE_PAYMENT_DATE | INVOICE_ID | PAID_BY |
|---|--------------------|-------------------------|-------------------------|------------|---------|
| 1 | 400008 | 50.00 | 2016-03-26 00:00:00.000 | 101003 | 1008 |
| 2 | 400002 | 5000.00 | 2015-02-26 00:00:00.000 | 101007 | 1008 |
| 3 | 400003 | 4100.00 | 2016-05-26 00:00:00.000 | 101000 | 1016 |
| 4 | 400004 | 1500.00 | 2016-05-01 00:00:00.000 | 101000 | 1016 |
| 5 | 400005 | 2000.00 | 2016-07-10 00:00:00.000 | 101008 | 1016 |
| 6 | 400008 | 10000.00 | 2016-08-10 00:00:00.000 | 101012 | 1016 |
| 7 | 400007 | 500.00 | 2015-07-01 00:00:00.000 | 101013 | 1006 |
| 8 | 400009 | 14100.00 | 2016-05-12 00:00:00.000 | 101016 | 1016 |

Business Process 3:

Employees create reimbursement requests for any official expenses like travel, food, accommodation etc. which is assigned to procurement and payment associate for review. For approved requests, the procurement and payment associate pays the employee.

Step 1: Reimbursement request creation

BEFORE PROCESSING THE QUERY

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The toolbar at the top includes standard options like Tools, Window, Help, and various execution and navigation icons. Below the toolbar, the title bar displays multiple open queries: 'SQLQuery16.sql - ...HABH\Rishabh (60)*', 'SQLQuery15.sql - ...HABH\Rishabh (58)*' (which is the active query window), 'SQLQuery10.sql - ...HABH\Rishabh (56)*', and 'SQLQuery1.sql - RI...HABH\Rishabh (52)*'. The main area contains a query editor window with the following SQL code:

```
SELECT*
FROM REIMBURSEMENT REQUEST
```

Below the query editor is a results grid titled 'Results'. The grid displays 19 rows of data from the 'REIMBURSEMENT REQUEST' table. The columns are: REIMBURSEMENT_REQUEST_ID, REQUESTOR_ID, ASSIGNED_TO_ID, SUBMITTED_DATE, REIMBURSEMENT_REQUEST_STATUS, DEPARTMENT, and WORK_LOCATION. The data is as follows:

| REIMBURSEMENT_REQUEST_ID | REQUESTOR_ID | ASSIGNED_TO_ID | SUBMITTED_DATE | REIMBURSEMENT_REQUEST_STATUS | DEPARTMENT | WORK_LOCATION |
|--------------------------|--------------|----------------|-------------------------|------------------------------|------------|---------------|
| 1 | E001 | 1000 | 2015-06-16 00:00:00.000 | OPEN | Marketing | Santa Clara |
| 2 | E002 | 1000 | 2015-06-09 00:00:00.000 | OPEN | Marketing | Santa Clara |
| 3 | E003 | 1001 | 2015-12-29 00:00:00.000 | OPEN | Technology | Milpitas |
| 4 | E005 | 1003 | 2015-08-09 00:00:00.000 | OPEN | Technology | Santa Clara |
| 5 | E006 | 1005 | 2016-05-16 00:00:00.000 | OPEN | Technology | Milpitas |
| 6 | E007 | 1005 | 2016-01-28 00:00:00.000 | OPEN | Technology | Milpitas |
| 7 | E008 | 1006 | 2015-06-14 00:00:00.000 | APPROVED | Accounting | New York |
| 8 | E009 | 1003 | 2015-12-01 00:00:00.000 | APPROVED | Accounting | New York |
| 9 | E010 | 1007 | 2015-12-11 00:00:00.000 | APPROVED | Technology | New York |
| 10 | E011 | 1007 | 2016-04-09 00:00:00.000 | APPROVED | Technology | New York |
| 11 | E012 | 1008 | 2016-05-14 00:00:00.000 | APPROVED | Accounting | San Francisco |
| 12 | E013 | 1000 | 2015-10-04 00:00:00.000 | APPROVED | Production | Milpitas |
| 13 | E014 | 1010 | 2016-02-10 00:00:00.000 | APPROVED | Technology | New York |
| 14 | E015 | 1011 | 2016-03-11 00:00:00.000 | APPROVED | Technology | Milpitas |
| 15 | E016 | 1012 | 2016-03-07 00:00:00.000 | REJECTED | Sales | Milpitas |
| 16 | E017 | 1013 | 2015-09-13 00:00:00.000 | REJECTED | Technology | San Francisco |
| 17 | E018 | 1015 | 2016-01-04 00:00:00.000 | REJECTED | Technology | New York |
| 18 | E019 | 1017 | 2015-10-15 00:00:00.000 | REJECTED | Admin | San Francisco |
| 19 | E020 | 1018 | 2016-04-17 00:00:00.000 | REJECTED | Marketing | San Diego |

AFTER PROCESSING THE QUERY:

iIS-2603-Project-Expensify (RISHABH\Rishabh (60)) - Microsoft SQL Server Management Studio

Tools Window Help

Execute Debug SQLQuery16.sql ...HABH\Rishabh (60)* SQLQuery15.sql ...HABH\Rishabh (58)* SQLQuery10.sql ...HABH\Rishabh (56)* SQLQuery1.sql - R...HABH\Rishabh (52)* SQLQuery12.sql ...HABH\Rishabh (57)*

```
INSERT INTO REIMBURSEMENT_REQUEST(REQUESTOR_ID, ASSIGNED_TO, SUBMITTED_DATE, REIMBURSEMENT_REQUEST_STATUS, DEPARTMENT, WORK_LOCATION)
VALUES ('8021', '1027', '1008', '04/28/2016', 'Open', 'Accounting', 'San Francisco')

SELECT*
FROM REIMBURSEMENT_REQUEST
```

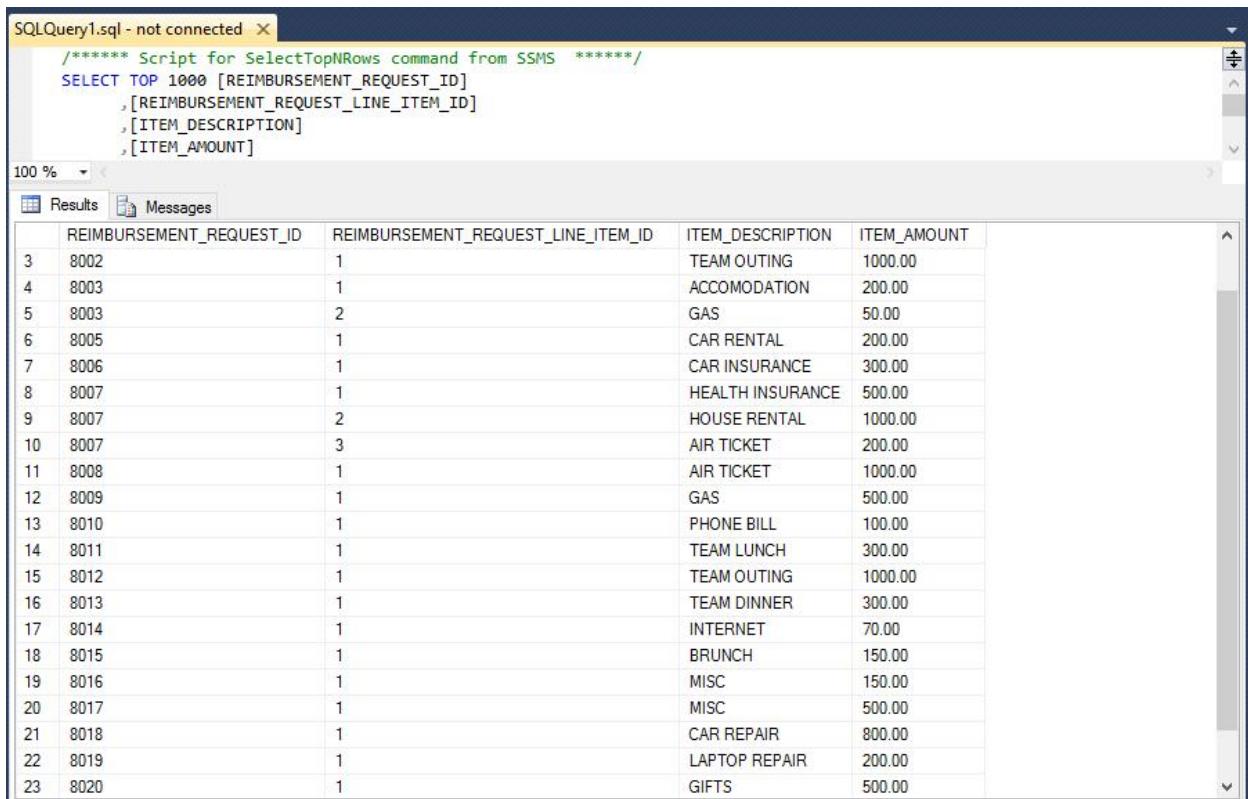
100 % < >

Results Messages

| REIMBURSEMENT_REQUEST_ID | REQUESTOR_ID | ASSIGNED_TO | SUBMITTED_DATE | REIMBURSEMENT_REQUEST_STATUS | DEPARTMENT | WORK_LOCATION | |
|--------------------------|--------------|-------------|----------------|------------------------------|------------|---------------|---------------|
| 1 | 8001 | 1000 | 1006 | 2015-06-16 00:00:00.000 | OPEN | Marketing | Santa Clara |
| 2 | 8002 | 1000 | 1008 | 2015-06-09 00:00:00.000 | OPEN | Marketing | Santa Clara |
| 3 | 8003 | 1001 | 1016 | 2015-12-29 00:00:00.000 | OPEN | Technology | Milpitas |
| 4 | 8005 | 1003 | 1006 | 2015-08-09 00:00:00.000 | OPEN | Technology | Santa Clara |
| 5 | 8006 | 1005 | 1006 | 2016-05-16 00:00:00.000 | OPEN | Technology | Milpitas |
| 6 | 8007 | 1005 | 1008 | 2016-01-28 00:00:00.000 | OPEN | Technology | Milpitas |
| 7 | 8008 | 1006 | 1016 | 2015-06-14 00:00:00.000 | APPROVED | Accounting | New York |
| 8 | 8009 | 1006 | 1008 | 2015-12-01 00:00:00.000 | APPROVED | Accounting | New York |
| 9 | 8010 | 1007 | 1006 | 2015-12-11 00:00:00.000 | APPROVED | Technology | New York |
| 10 | 8011 | 1007 | 1016 | 2016-04-09 00:00:00.000 | APPROVED | Technology | New York |
| 11 | 8012 | 1008 | 1006 | 2016-05-14 00:00:00.000 | APPROVED | Accounting | San Francisco |
| 12 | 8013 | 1009 | 1008 | 2015-10-04 00:00:00.000 | APPROVED | Production | Milpitas |
| 13 | 8014 | 1010 | 1006 | 2016-02-10 00:00:00.000 | APPROVED | Technology | New York |
| 14 | 8015 | 1011 | 1016 | 2016-03-11 00:00:00.000 | APPROVED | Technology | Milpitas |
| 15 | 8016 | 1012 | 1008 | 2016-03-07 00:00:00.000 | REJECTED | Sales | Milpitas |
| 16 | 8017 | 1013 | 1006 | 2015-09-13 00:00:00.000 | REJECTED | Technology | San Francisco |
| 17 | 8018 | 1015 | 1016 | 2016-01-04 00:00:00.000 | REJECTED | Technology | New York |
| 18 | 8019 | 1017 | 1006 | 2015-10-15 00:00:00.000 | REJECTED | Admin | San Francisco |
| 19 | 8020 | 1018 | 1016 | 2016-04-17 00:00:00.000 | REJECTED | Marketing | San Diego |
| 20 | 8021 | 1027 | 1008 | 2016-05-01 00:00:00.000 | OPEN | Accounting | San Francisco |

Insert Reimbursement Request Line Items

BEFORE EXECUTING THE QUERY



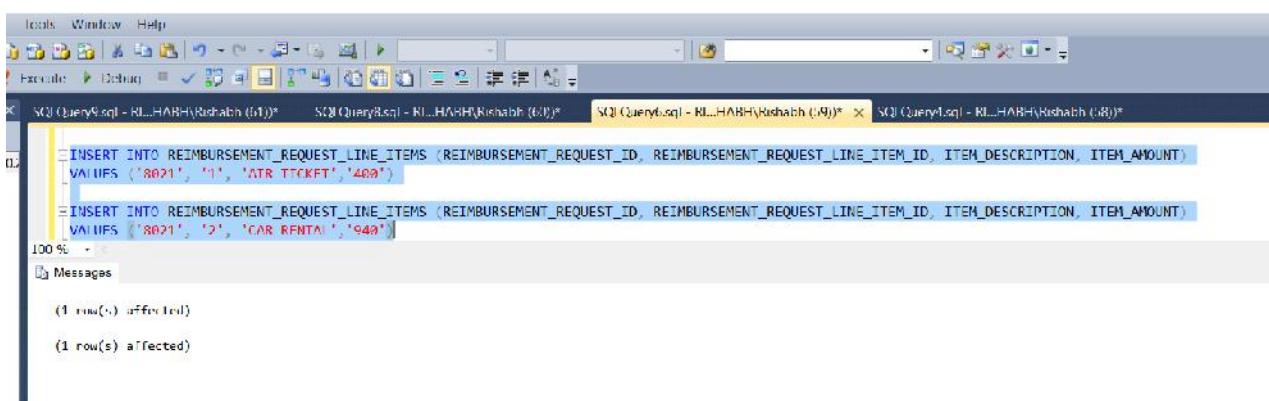
The screenshot shows the SSMS interface with a query window titled "SQLQuery1.sql - not connected". The query displayed is:

```
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [REIMBURSEMENT_REQUEST_ID]
      ,[REIMBURSEMENT_REQUEST_LINE_ITEM_ID]
      ,[ITEM_DESCRIPTION]
      ,[ITEM_AMOUNT]
```

The results grid displays 23 rows of data from the REIMBURSEMENT_REQUEST_LINE_ITEMS table. The columns are:

| | REIMBURSEMENT_REQUEST_ID | REIMBURSEMENT_REQUEST_LINE_ITEM_ID | ITEM_DESCRIPTION | ITEM_AMOUNT |
|----|--------------------------|------------------------------------|------------------|-------------|
| 3 | 8002 | 1 | TEAM OUTING | 1000.00 |
| 4 | 8003 | 1 | ACCOMODATION | 200.00 |
| 5 | 8003 | 2 | GAS | 50.00 |
| 6 | 8005 | 1 | CAR RENTAL | 200.00 |
| 7 | 8006 | 1 | CAR INSURANCE | 300.00 |
| 8 | 8007 | 1 | HEALTH INSURANCE | 500.00 |
| 9 | 8007 | 2 | HOUSE RENTAL | 1000.00 |
| 10 | 8007 | 3 | AIR TICKET | 200.00 |
| 11 | 8008 | 1 | AIR TICKET | 1000.00 |
| 12 | 8009 | 1 | GAS | 500.00 |
| 13 | 8010 | 1 | PHONE BILL | 100.00 |
| 14 | 8011 | 1 | TEAM LUNCH | 300.00 |
| 15 | 8012 | 1 | TEAM OUTING | 1000.00 |
| 16 | 8013 | 1 | TEAM DINNER | 300.00 |
| 17 | 8014 | 1 | INTERNET | 70.00 |
| 18 | 8015 | 1 | BRUNCH | 150.00 |
| 19 | 8016 | 1 | MISC | 150.00 |
| 20 | 8017 | 1 | MISC | 500.00 |
| 21 | 8018 | 1 | CAR REPAIR | 800.00 |
| 22 | 8019 | 1 | LAPTOP REPAIR | 200.00 |
| 23 | 8020 | 1 | GIFTS | 500.00 |

QUERY PROCESSING



The screenshot shows the SSMS interface with four tabs: "SQL Query1.sql - RL...HABH(Rishabh (61))", "SQL Query2.sql - RL...HABH(Rishabh (60))", "SQL Query3.sql - RL...HABH(Rishabh (69))", and "SQL Query4.sql - RL...HABH(Rishabh (68))". The active tab contains the following SQL code:

```
INSERT INTO REIMBURSEMENT_REQUEST_LINE_ITEMS (REIMBURSEMENT_REQUEST_ID, REIMBURSEMENT_REQUEST_LINE_ITEM_ID, ITEM_DESCRIPTION, ITEM_AMOUNT)
VALUES ('8021', '1', 'AIR TICKET', '400')

INSERT INTO REIMBURSEMENT_REQUEST_LINE_ITEMS (REIMBURSEMENT_REQUEST_ID, REIMBURSEMENT_REQUEST_LINE_ITEM_ID, ITEM_DESCRIPTION, ITEM_AMOUNT)
VALUES ('8021', '2', 'CAR RENTAL', '940')
```

The "Messages" pane at the bottom shows the results of the execution:

```
(1 row(s) affected)

(1 row(s) affected)
```

Step 2: Reimbursement payment

```
INSERT INTO REIMBURSEMENT_PAYMENT
(REIMBURSEMENT_PAYMENT_ID,REIMBURSEMENT_PAYMENT_AMOUNT,
REIMBURSEMENT_PAYMENT_DATE,
REIMBURSEMENT_REQUEST_ID,PAID_BY)
VALUES('901002','1340','2016-05-20','8021','1016');
```

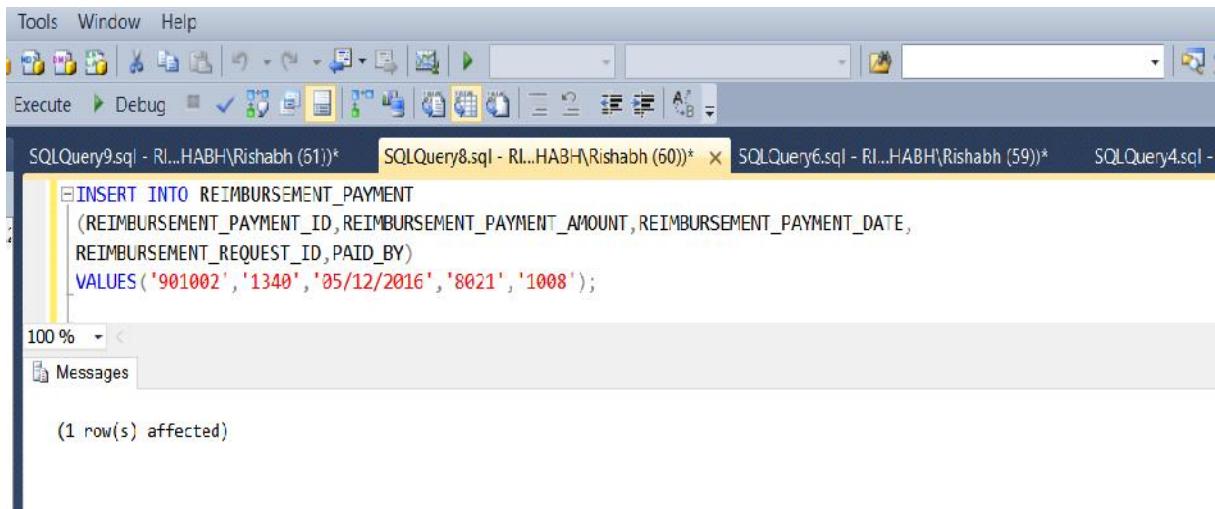
The screenshot shows a SQL Server Management Studio window with the following details:

- Tab bar: SQLQuery32.sql - D...I33MQST\rai7s (62)*, DESKTOP-I33MQST\RAI7S...REIMENT_PAYMENT, DESKTOP-I33MQST...INVOICE_PAYMENT
- Query pane: SELECT * FROM REIMBURSEMENT_PAYMENT;
- Results pane: Displays a table with 8 rows of data.
- Table columns: REIMBURSEMENT_PAYMENT_ID, REIMBURSEMENT_PAYMENT_AMOUNT, REIMBURSEMENT_PAYMENT_DATE, REIMBURSEMENT_REQUEST_ID, PAID_BY.
- Data rows:

| | REIMBURSEMENT_PAYMENT_ID | REIMBURSEMENT_PAYMENT_AMOUNT | REIMBURSEMENT_PAYMENT_DATE | REIMBURSEMENT_REQUEST_ID | PAID_BY |
|---|--------------------------|------------------------------|----------------------------|--------------------------|---------|
| 1 | 90102 | 500.00 | 2015-12-20 00:00:00.000 | 8009 | 1008 |
| 2 | 90103 | 100.00 | 2015-12-30 00:00:00.000 | 8010 | 1006 |
| 3 | 90104 | 300.00 | 2016-04-13 00:00:00.000 | 8011 | 1016 |
| 4 | 90105 | 1000.00 | 2016-05-26 00:00:00.000 | 8012 | 1006 |
| 5 | 90106 | 300.00 | 2015-10-30 00:00:00.000 | 8013 | 1008 |
| 6 | 90107 | 70.00 | 2016-02-20 00:00:00.000 | 8014 | 1006 |
| 7 | 90108 | 150.00 | 2016-03-30 00:00:00.000 | 8015 | 1016 |
| 8 | 901001 | 1000.00 | 2015-06-20 00:00:00.000 | 8008 | 1016 |

- Message bar: Query executed successfully. DESKTOP-I33MQST\SQLEXPRESS ... DESKTOP-I33MQST\rai7s ... MSIS-2603-Project-Expe... 00:00:00 8 rows

QUERY PROCESSING FOR REIMBURSEMENT PAYMENT

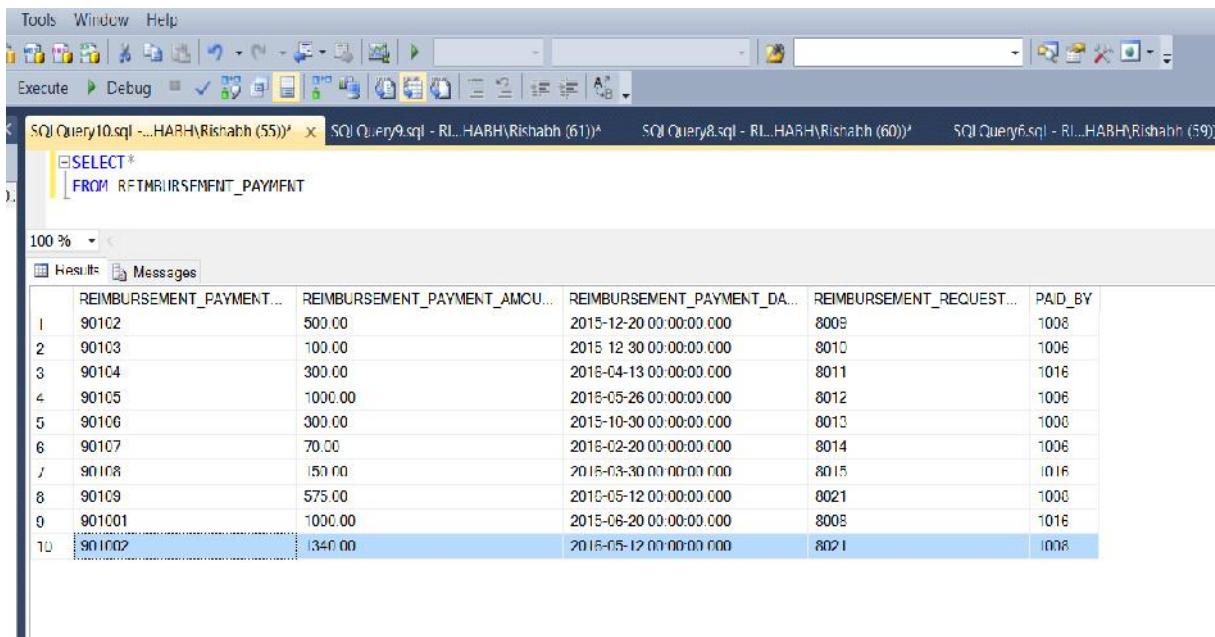


The screenshot shows the SQL Server Management Studio interface. A query window is open with the following SQL code:

```
INSERT INTO REIMBURSEMENT_PAYMENT
  (REIMBURSEMENT_PAYMENT_ID, REIMBURSEMENT_PAYMENT_AMOUNT, REIMBURSEMENT_PAYMENT_DATE,
  REIMBURSEMENT_REQUEST_ID, PAID_BY)
VALUES ('901002', '1340', '05/12/2016', '8021', '1008');
```

The status bar at the bottom indicates "(1 row(s) affected)".

AFTER QUERY PROCESSING FOR REIMBURSEMENT PAYMENT



The screenshot shows the SQL Server Management Studio interface after the query has been processed. A query window is open with the following SQL code:

```
SELECT *
FROM REIMBURSEMENT_PAYMENT
```

The results pane displays the following data:

| | REIMBURSEMENT_PAYMENT_ID | REIMBURSEMENT_PAYMENT_AMOUNT | REIMBURSEMENT_PAYMENT_DATE | REIMBURSEMENT_REQUEST_ID | PAID_BY |
|----|--------------------------|------------------------------|----------------------------|--------------------------|---------|
| 1 | 90102 | 500.00 | 2015-12-20 00:00:00.000 | 8009 | 1008 |
| 2 | 90103 | 100.00 | 2015-12-30 00:00:00.000 | 8010 | 1006 |
| 3 | 90104 | 300.00 | 2016-04-13 00:00:00.000 | 8011 | 1016 |
| 4 | 90105 | 1000.00 | 2015-05-26 00:00:00.000 | 8012 | 1006 |
| 5 | 90106 | 300.00 | 2015-10-30 00:00:00.000 | 8013 | 1000 |
| 6 | 90107 | 70.00 | 2016-02-20 00:00:00.000 | 8014 | 1006 |
| 7 | 90108 | 150.00 | 2016-03-30 00:00:00.000 | 8015 | 1016 |
| 8 | 90109 | 575.00 | 2016-05-12 00:00:00.000 | 8021 | 1000 |
| 9 | 901001 | 1000.00 | 2015-06-20 00:00:00.000 | 8008 | 1016 |
| 10 | 901002 | 1340.00 | 2016-05-12 00:00:00.000 | 8021 | 1008 |

Additional Queries

1. Find all WIP purchase orders assigned a particular supplier

```
SELECT * FROM PURCHASE_ORDER  
WHERE ORDER_STATUS = 'WIP'  
AND ASSIGNED_TO IN (  
SELECT SUPPLIER_ID FROM SUPPLIER WHERE SUPPLIER_NAME = 'Kayveo')
```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql - DESKTOP-1B3MQST\ra7s (54)*' contains the following T-SQL code:

```
SELECT * FROM PURCHASE_ORDER  
WHERE ORDER_STATUS = 'WIP'  
AND ASSIGNED_TO IN(SELECT SUPPLIER_ID FROM SUPPLIER WHERE SUPPLIER_NAME = 'Kayveo')
```

The results pane shows one row of data:

| ORDER_ID | PURCHASE_REQUEST_ID | ORDER_CREATED_DATE | ORDER_FULFILLMENT_DATE | CREATED_BY | ASSIGNED_TO | DEPARTMENT | WORK_LOCATION | ORDER_STATUS | ORDER_COMPLETED_DATE |
|----------|---------------------|-------------------------|------------------------|------------|-------------|------------|---------------|--------------|----------------------|
| 512 | 423 | 2016-04-10 00:00:00.000 | NULL | 108 | 206 | Sales | Mipitas | WIP | NULL |

2. View all requests that have been approved for first quarter of 2016

```
SELECT *  
FROM PURCHASE_REQUEST PR  
WHERE PR.REQUEST_STATUS = 'Approved' AND  
PR.APPROVER REVIEW_DATE BETWEEN '01/01/2016' AND '03/31/2016'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql - DEJ3EMQST\ra7s (54)*' contains the following T-SQL code:

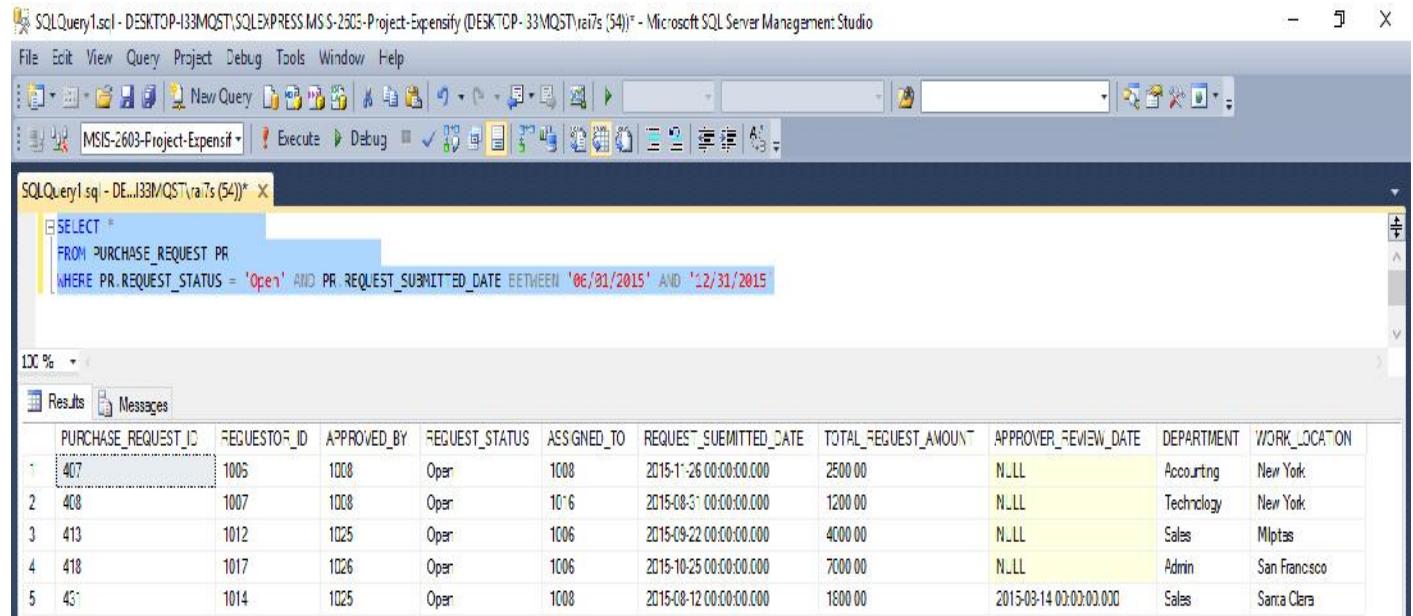
```
SELECT *  
FROM PURCHASE_REQUEST PR  
WHERE PR.REQUEST_STATUS = 'Approved' AND PR.APPROVER REVIEW_DATE BETWEEN '01/01/2016' AND '03/31/2016'
```

The results pane shows six rows of data:

| PURCHASE_REQUEST_ID | REQUESTOR_ID | APPROVED_BY | REQUEST_STATUS | ASSIGNED_TO | REQUEST_SUBMITTED_DATE | TOTAL_REQUEST_AMOUNT | APPROVER REVIEW_DATE | DEPARTMENT | WORK_LOCATION |
|---------------------|--------------|-------------|----------------|-------------|-------------------------|----------------------|-------------------------|------------|---------------|
| 404 | 1003 | 1023 | Approved | 105 | 2016-03-19 00:00:00.000 | 2000.00 | 2016-03-22 00:00:00.000 | Technology | Santa Clara |
| 406 | 1005 | 1003 | Approved | 1005 | 2016-02-09 00:00:00.000 | 2000.00 | 2016-02-10 00:00:00.000 | Technology | Mipitas |
| 409 | 1008 | 1022 | Approved | 1003 | 2016-02-18 00:00:00.000 | 8000.00 | 2016-02-20 00:00:00.000 | Accounting | San Francisco |
| 424 | 1008 | 1022 | Approved | 1005 | 2016-03-08 00:00:00.000 | 400.00 | 2016-03-10 00:00:00.000 | Accounting | San Francisco |
| 428 | 1012 | 1025 | Approved | 105 | 2016-01-18 00:00:00.000 | 2000.00 | 2016-01-19 00:00:00.000 | Sales | Mipitas |
| 432 | 1017 | 1025 | Approved | 1005 | 2016-02-10 00:00:00.000 | 6000.00 | 2016-02-10 00:00:00.000 | Accts | San Francisco |

3. View all requests from 2015 that are still pending approval

```
SELECT *
FROM PURCHASE_REQUEST PR
WHERE PR.REQUEST_STATUS = 'Open' AND
PR.REQUEST_SUBMITTED_DATE BETWEEN '06/01/2015' AND '12/31/2015'
```



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql - DESKTOP-I33MQST\SQLExpress MSS-2503-Project-Expensify (DESKTOP-I33MQST\ra7s (54))' is open. The query itself is:

```
SELECT *
FROM PURCHASE_REQUEST PR
WHERE PR.REQUEST_STATUS = 'Open' AND PR.REQUEST_SUBMITTED_DATE BETWEEN '06/01/2015' AND '12/31/2015'
```

The results grid displays five rows of data, each representing a purchase request. The columns are: PURCHASE_REQUEST_ID, REQUESTOR_ID, APPROVED_BY, REQUEST_STATUS, ASSIGNED_TO, REQUEST_SUBMITTED_DATE, TOTAL_REQUEST_AMOUNT, APPROVER REVIEW_DATE, DEPARTMENT, and WORK_LOCATION.

| | PURCHASE_REQUEST_ID | REQUESTOR_ID | APPROVED_BY | REQUEST_STATUS | ASSIGNED_TO | REQUEST_SUBMITTED_DATE | TOTAL_REQUEST_AMOUNT | APPROVER REVIEW_DATE | DEPARTMENT | WORK_LOCATION |
|---|---------------------|--------------|-------------|----------------|-------------|-------------------------|----------------------|-------------------------|------------|---------------|
| 1 | 407 | 1006 | 1008 | Open | 1008 | 2015-12-26 00:00:00.000 | 2500.00 | NULL | Accounting | New York |
| 2 | 408 | 1007 | 1008 | Open | 1016 | 2015-08-31 00:00:00.000 | 1200.00 | NULL | Technology | New York |
| 3 | 413 | 1012 | 1025 | Open | 1006 | 2015-09-22 00:00:00.000 | 4000.00 | NULL | Sales | Miptas |
| 4 | 418 | 1017 | 1026 | Open | 1006 | 2015-10-25 00:00:00.000 | 7000.00 | NULL | Admin | San Francisco |
| 5 | 431 | 1014 | 1025 | Open | 1008 | 2015-08-12 00:00:00.000 | 1800.00 | 2015-03-14 00:00:00.000 | Sales | Santa Clara |

4. Find all payments made by a payment associate in last quarter of 2015

```
SELECT I.INVOICE_PAYMENT_ID,I.INVOICE_PAYMENT_AMOUNT,R.REIMBURSEMENT_PAYMENT_ID,  
R.REIMBURSEMENT_PAYMENT_AMOUNT FROM INVOICE_PAYMENT I, REIMBURSEMENT_PAYMENT R  
WHERE (I.PAID_BY = R.PAID_BY)  
AND I.PAID_BY = 1016  
AND (I.INVOICE_PAYMENT_DATE BETWEEN '06/01/2015' AND '12/31/2015')  
AND (R.REIMBURSEMENT_PAYMENT_DATE BETWEEN '06/01/2015' AND '12/31/2015')
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the following SQL code:

```
SELECT I.INVOICE_PAYMENT_ID,I.INVOICE_PAYMENT_AMOUNT,R.REIMBURSEMENT_PAYMENT_ID,R.REIMBURSEMENT_PAYMENT_AMOUNT FROM INVOICE_PAYMENT I, REIMBURSEMENT_PAYMENT R  
WHERE (I.PAID_BY = R.PAID_BY)  
AND I.PAID_BY = 1016  
AND (I.INVOICE_PAYMENT_DATE BETWEEN '06/01/2015' AND '12/31/2015')  
AND (R.REIMBURSEMENT_PAYMENT_DATE BETWEEN '06/01/2015' AND '12/31/2015')
```

The results pane displays the following data:

| | INVOICE_PAYMENT_ID | INVOICE_PAYMENT_AMOUNT | REIMBURSEMENT_PAYMENT_ID | REIMBURSEMENT_PAYMENT_AMOUNT |
|---|--------------------|------------------------|--------------------------|------------------------------|
| 1 | 400005 | 2000.00 | 901001 | 1000.00 |
| 2 | 400006 | 1000.00 | 901001 | 1000.00 |

5. Find all overdue payments

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the following SQL code:

```
SELECT P.INVOICE_PAYMENT_ID,P.INVOICE_ID,I.ORDER_ID FROM INVOICE I,INVOICE_PAYMENT P  
WHERE I.INVOICE_ID=P.INVOICE_ID  
AND P.INVOICE_PAYMENT_DATE>I.PAYMENT_DUE_DATE
```

The results pane displays the following data:

| | INVOICE_PAYMENT_ID | INVOICE_ID | ORDER_ID |
|---|--------------------|------------|----------|
| 1 | 400002 | 101007 | 510 |
| 2 | 400004 | 101009 | 514 |

8. Business Metrics Queries

1. Find Suppliers by Total Spend

```
/*Group suppliers by spend*/
SELECT SUPPLIER.SUPPLIER_NAME,SUM(INVOICE_PAYMENT.INVOICE_PAYMENT_AMOUNT ) AS AMOUNT_SPENT
FROM SUPPLIER,INVOICE,INVOICE_PAYMENT
WHERE INVOICE_PAYMENT.INVOICE_ID = INVOICE.INVOICE_ID AND SUPPLIER.SUPPLIER_ID = INVOICE.SUPPLIER_ID
GROUP BY SUPPLIER.SUPPLIER_NAME
```

100 % < Results Messages

| | SUPPLIER_NAME | AMOUNT_SPENT |
|---|---------------|--------------|
| 1 | InnoZ | 4100.00 |
| 2 | Kayveo | 5000.00 |
| 3 | Novatech | 50.00 |
| 4 | Skipstorm | 500.00 |
| 5 | Skyble | 13500.00 |

2. Find the total spend with a drill down of Reimbursement payments and Procurement/Supplier payments by each department and work location

```
SELECT O.DEPARTMENT,O.WORK_LOCATION,SUM(IP.INVOICE_PAYMENT_AMOUNT) AS 'SUPPLIER PAYMENT',
SUM(RP.REIMBURSEMENT_PAYMENT_AMOUNT) AS 'REIMBURSEMENT PAYMENT',
SUM(IP.INVOICE_PAYMENT_AMOUNT + RP.REIMBURSEMENT_PAYMENT_AMOUNT) AS TOTAL_SPEND
FROM PURCHASE_ORDER O,INVOICE_PAYMENT IP,INVOICE I,REIMBURSEMENT_REQUEST R,REIMBURSEMENT_PAYMENT RP
WHERE IP.INVOICE_ID = I.INVOICE_ID AND I.ORDER_ID = O.ORDER_ID
AND RP.REIMBURSEMENT_REQUEST_ID = R.REIMBURSEMENT_REQUEST_ID
GROUP BY O.DEPARTMENT,O.WORK_LOCATION
ORDER BY O.WORK_LOCATION
```

100 % < Results Messages

| | DEPARTMENT | WORK_LOCATION | SUPPLIER PAYMENT | REIMBURSEMENT PAYMENT | TOTAL_SPEND |
|---|------------|---------------|------------------|-----------------------|-------------|
| 1 | Sales | Milpitas | 13500.00 | 3995.00 | 17495.00 |
| 2 | Technology | Milpitas | 127350.00 | 11985.00 | 139335.00 |
| 3 | Accounting | New York | 4500.00 | 3995.00 | 8495.00 |
| 4 | Marketing | San Diego | 45000.00 | 3995.00 | 48995.00 |
| 5 | Sales | Santa Clara | 18000.00 | 3995.00 | 21995.00 |

3. List Suppliers who have delayed their deliveries along with the duration for delay in days

```

SELECT A.SUPPLIER,S.SUPPLIER_NAME,A.PURCHASE_ORDER_ID, (A.DELAY_IN_DAYS - 20) AS 'DELAY_IN_DAYS' ,A.ORDER_STATUS
FROM
(SELECT PO.ORDER_ID AS 'PURCHASE_ORDER_ID',PO.ASSIGNED_TO AS 'SUPPLIER',
DATEDIFF(DAY,PO.ORDER_CREATED_DATE,PO.ORDER_FULFILLMENT_DATE)
AS 'DELAY_IN_DAYS', PO.ORDER_STATUS FROM PURCHASE_ORDER PO WHERE PO.ORDER_STATUS = 'COMPLETED' AND EXISTS
(SELECT * FROM PURCHASE_ORDER P WHERE DATEDIFF(DAY,P.ORDER_CREATED_DATE,P.ORDER_FULFILLMENT_DATE)>20 AND
PO.ORDER_ID = P.ORDER_ID) A,SUPPLIER S
WHERE S.SUPPLIER_ID = A.SUPPLIER |

```

| | SUPPLIER | SUPPLIER_NAME | PURCHASE_ORDER_ID | DELAY_IN_DAYS | ORDER_STATUS |
|---|----------|---------------|-------------------|---------------|--------------|
| 1 | 210 | Skyble | 508 | 2 | COMPLETED |
| 2 | 206 | Kayveo | 510 | 8 | COMPLETED |
| 3 | 210 | Skyble | 514 | 68 | COMPLETED |
| 4 | 210 | Skyble | 518 | 6 | COMPLETED |

4. Find the Suppliers for which the quantity delivered was short of quantity ordered

```

/*Find the suppliers for which the quantity delivered was short of quantity ordered.*/
SELECT SUPPLIER.SUPPLIER_NAME + '-' + INVOICE_LINE_ITEMS.ITEM_DESCRIPTION AS 'SUPPLIER-ITEM',
INVOICE_LINE_ITEMS.ITEM_QUANTITY_ORDERED,INVOICE_LINE_ITEMS.ITEM_QUANTITY_DELIVERED
FROM PURCHASE_ORDER,INVOICE, INVOICE_LINE_ITEMS, SUPPLIER
WHERE PURCHASE_ORDER.ORDER_ID = INVOICE.ORDER_ID AND
INVOICE.INVOICE_ID = INVOICE_LINE_ITEMS.INVOICE_ID AND
INVOICE_LINE_ITEMS.ITEM_QUANTITY_DELIVERED < INVOICE_LINE_ITEMS.ITEM_QUANTITY_ORDERED AND
PURCHASE_ORDER.ASSIGNED_TO = SUPPLIER.SUPPLIER_ID

```

| | SUPPLIER-ITEM | ITEM_QUANTITY_ORDERED | ITEM_QUANTITY_DELIVERED |
|---|------------------------|-----------------------|-------------------------|
| 1 | InnoZ-LENOVO SPEAKERS | 25 | 22 |
| 2 | Skipstom-APPLE LAPTOP | 25 | 20 |
| 3 | Nlounge-SAMSUNG TAB | 5 | 4 |
| 4 | Skipstom-APPLE MOUSE | 15 | 12 |
| 5 | Nlounge-SAMSUNG WEBCAM | 10 | 8 |
| 6 | InnoZ-LENOVO SPEAKERS | 12 | 10 |

5. Find the items purchased from different suppliers along with their quantity and the price the supplier charged for every item

```
/* price analysis*/
SELECT S.SUPPLIER_NAME, IL.ITEM_DESCRIPTION, IL.ITEM_PRICE, IL.ITEM_QUANTITY_ORDERED
FROM INVOICE I, INVOICE_LINE_ITEMS IL, SUPPLIER S
WHERE I.INVOICE_ID = IL.INVOICE_ID AND S.SUPPLIER_ID = I.SUPPLIER_ID
ORDER BY S.SUPPLIER_NAME
```

100 %

Results Messages

| | SUPPLIER_NAME | ITEM_DESCRIPTION | ITEM_PRICE | ITEM_QUANTITY_ORDERED |
|----|---------------|-------------------|------------|-----------------------|
| 1 | InnoZ | LENOVO MOUSE | 50.00 | 2 |
| 2 | InnoZ | LENOVO SPEAKERS | 200.00 | 25 |
| 3 | InnoZ | LENOVO MONITOR | 250.00 | 7 |
| 4 | InnoZ | LENOVO SPEAKERS | 200.00 | 8 |
| 5 | InnoZ | LENOVO MONITOR | 250.00 | 10 |
| 6 | InnoZ | LENOVO LAPTOP | 800.00 | 6 |
| 7 | InnoZ | LENOVO MOUSE | 50.00 | 8 |
| 8 | InnoZ | LENOVO SPEAKERS | 200.00 | 12 |
| 9 | Kayveo | SONY LAPTOP | 1000.00 | 5 |
| 10 | Nlounge | SAMSUNG WEBCAM | 800.00 | 5 |
| 11 | Nlounge | SAMSUNG TAB | 400.00 | 5 |
| 12 | Nlounge | SAMSUNG TAB | 400.00 | 5 |
| 13 | Nlounge | SAMSUNG WEBCAM | 800.00 | 10 |
| 14 | Novatech | LOGITECH KEYBOARD | 50.00 | 1 |
| 15 | Oodoo | MOTOROLA MODEM | 100.00 | 20 |
| 16 | Skipstorm | APPLE LAPTOP | 1000.00 | 4 |
| 17 | Skipstorm | APPLE MOUSE | 100.00 | 15 |
| 18 | Skipstorm | APPLE MONITOR | 1200.00 | 20 |
| 19 | Skipstorm | TP LINK ROUTER | 100.00 | 5 |
| 20 | Skipstorm | APPLE LAPTOP | 1000.00 | 25 |
| 21 | Skipstorm | APPLE MONITOR | 1200.00 | 3 |
| 22 | Skyble | TP LINK ROUTER | 100.00 | 20 |
| 23 | Skyble | TP LINK ROUTER | 100.00 | 15 |
| 24 | Skyble | APPLE LAPTOP | 1000.00 | 10 |

6. Find the status of purchase orders in every department since January 2015

```
/* Status of purchase orders in every department*/
SELECT O.DEPARTMENT,O.ORDER_STATUS,COUNT(O.ORDER_STATUS) AS 'Number of Orders'
FROM PURCHASE_ORDER O
WHERE O.ORDER_CREATED_DATE BETWEEN '01/01/2015' AND '05/30/2016'
GROUP BY O.DEPARTMENT,O.ORDER_STATUS
ORDER BY O.DEPARTMENT
```

100 % <

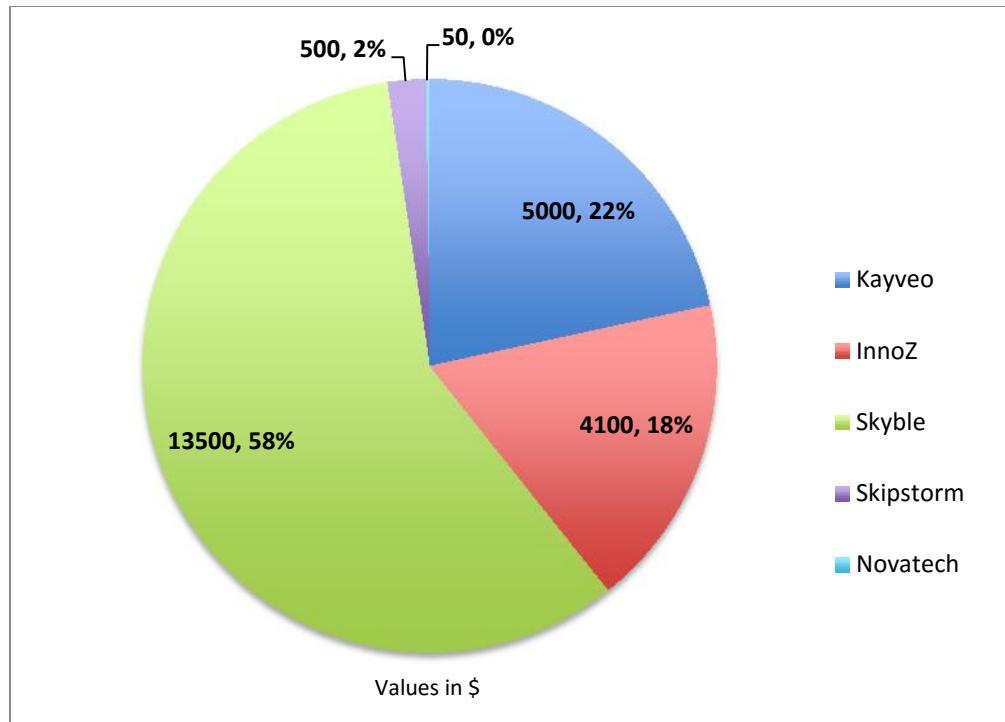
Results Messages

| | DEPARTMENT | ORDER_STATUS | Number of Orders |
|----|------------|--------------|------------------|
| 1 | Accounting | COMPLETED | 2 |
| 2 | Accounting | FULFILLED | 1 |
| 3 | Accounting | OPEN | 1 |
| 4 | Admin | FULFILLED | 1 |
| 5 | Admin | OPEN | 1 |
| 6 | Marketing | COMPLETED | 2 |
| 7 | Marketing | OPEN | 1 |
| 8 | Production | WIP | 1 |
| 9 | Sales | COMPLETED | 2 |
| 10 | Sales | FULFILLED | 2 |
| 11 | Sales | WIP | 1 |
| 12 | Technology | COMPLETED | 3 |
| 13 | Technology | FULFILLED | 2 |
| 14 | Technology | WIP | 1 |

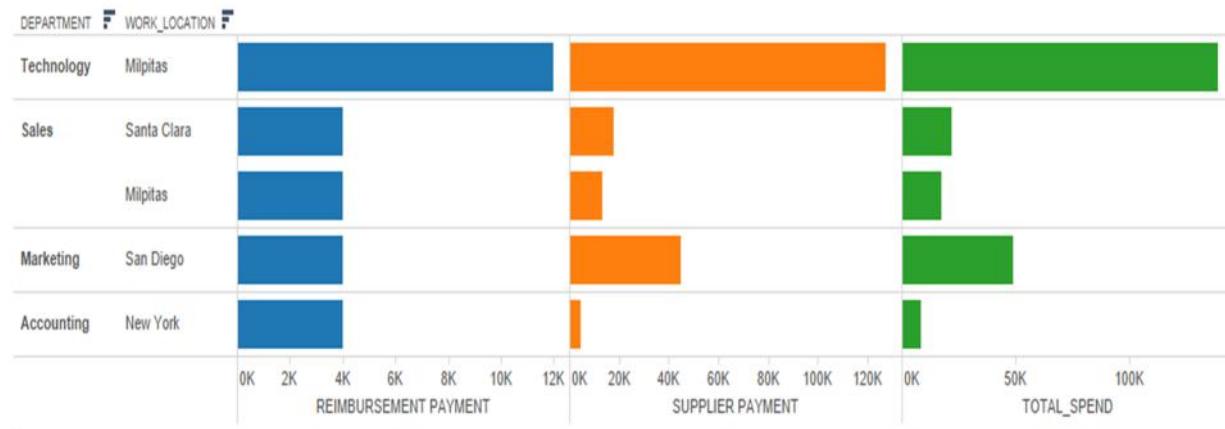
9. Business Metrics Reports

In today's world, with different systems in place a lot of data is being generated and it's very important to draw quantitative and qualitative insights from that data in the form of *Business Metrics Reports*, to help business take informed decisions.

Analyze the Percentage Spend by Supplier: This Report is crucial as it gives a holistic view about an organization's spending in \$ value and its percentage of the total spend by supplier. Based on \$ value purchase with a supplier, the company can initiate some negotiation discussions. From the report below we see that the company has spent 58% of its total spend with supplier Skyble, so it can definitely negotiate the payment terms for future purchases.



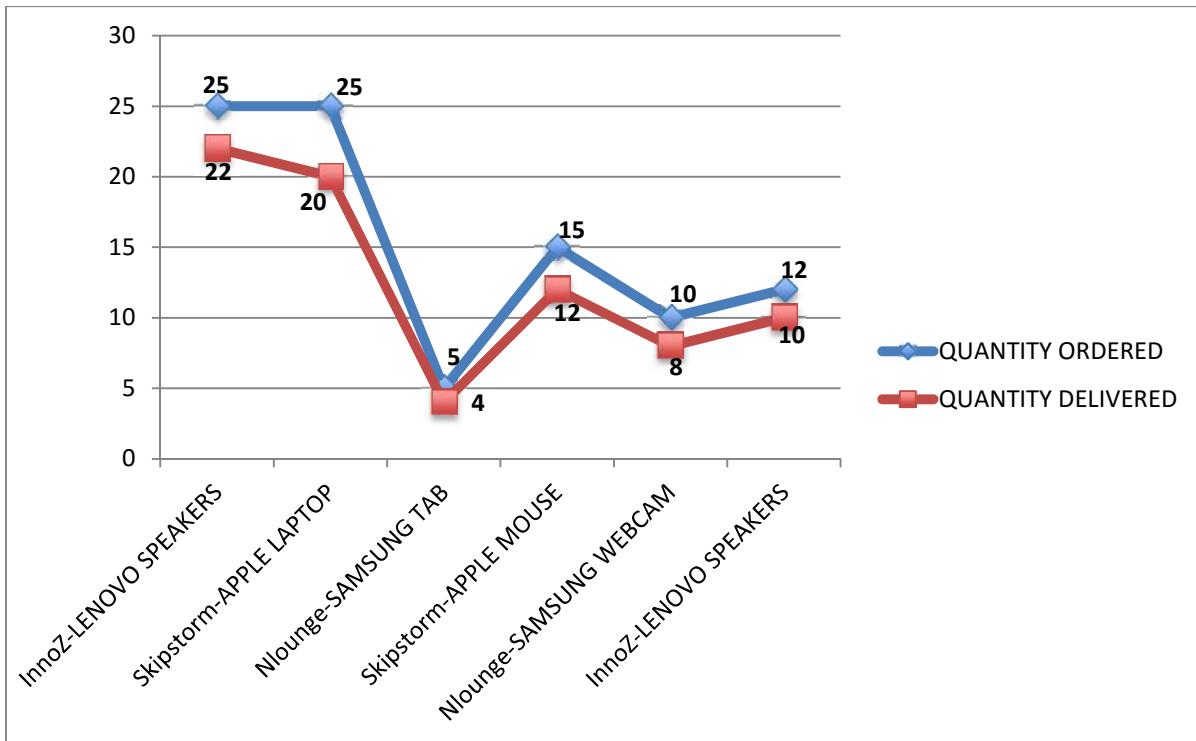
2. Analyze the Spends by Department and Work Location – This Report helps analyze the total spend with two *dimensions* – Department and work location *and it also drills down* to two subcategories of the total spend - reimbursement payments and procurement payments. This report tells us which department and which work location is spending the most and it also helps do comparative analysis. For example - Based on the following data, we see that the Milpitas location has the highest overall spend and hence management would want to have a closer look at the spending and see any opportunities to cut down costs.



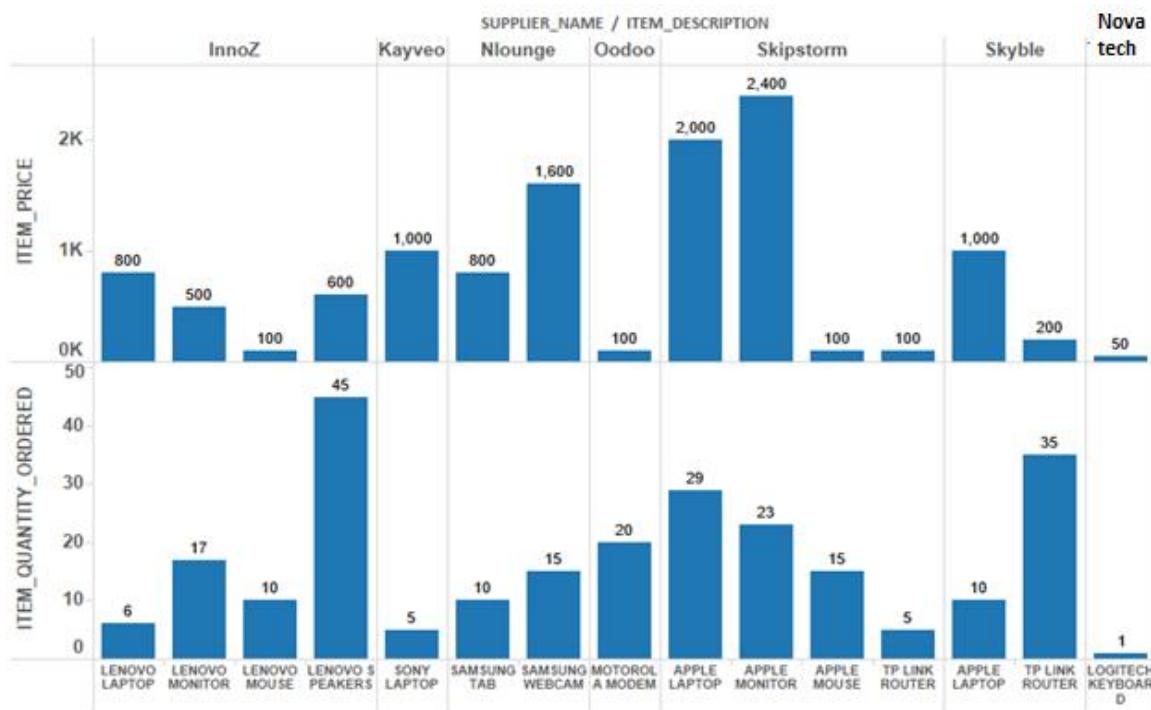
3. Suppliers with delays in delivery – This report shows the suppliers who have not delivered on time and also depicts the delay in days. From the report below we can infer that the supplier Skyble has not delivered on time for three orders, so probably, the company would not want to do business with this supplier in future.



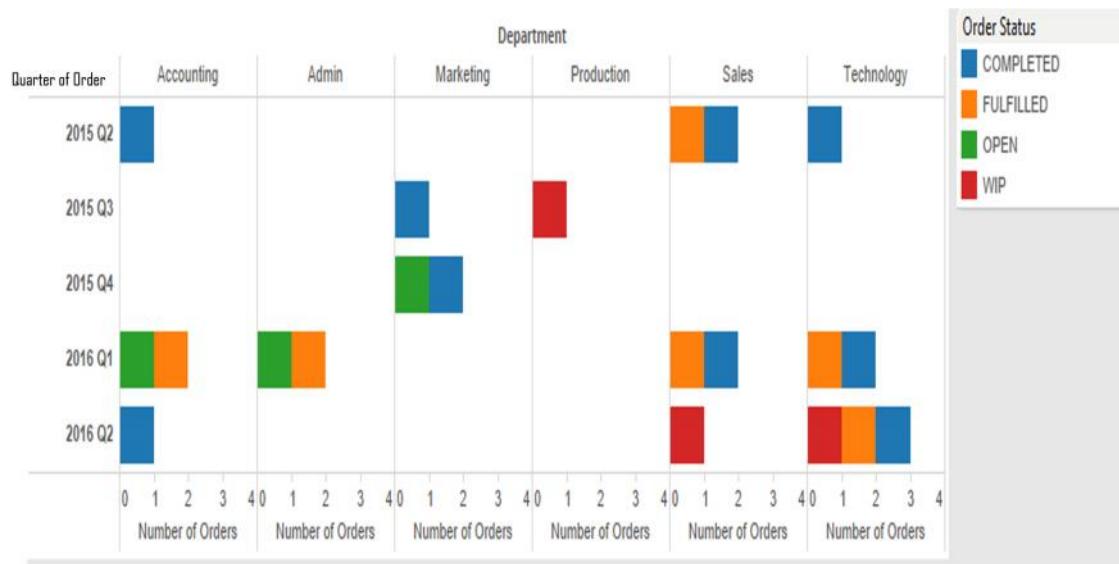
4. Delivery short on Quantity ordered – This report gives information of suppliers along with the Item names for which the quantity delivered was short of quantity ordered. This representation would help the procurement team to follow up with suppliers on delivery and also see the trend if a particular supplier is repeating this pattern of inefficiency over a period of time.



5. Items ordered and Item prices comparative analysis by supplier – This report helps analyze items ordered from different suppliers and their associated prices which gives an insight on the price variations for similar items. For example, apple laptops ordered are most expensive in laptop category, so the company could consider other options in the same category which are cost beneficial.



6. Purchase Order Status Analysis – This report allows analysis of purchase orders by their status. The procurement associate can keep track of orders created over time (view of last 6 quarters here) For example – Order created in Q3 -2015 is still in progress, so procurement team should prioritize this order and close it by following up with the associated supplier.



10. Project Summary

This project has helped us design and implement the back-end interactions of a procurement and expense management system, which has enabled us to understand the end-to-end activities of the system. As a team, we came together with varied perspectives and ideas for the project at hand and each one's skills and techniques were put into good use during the course of the project. High cohesiveness within the team helped us plan the project tasks ahead of time and also helped us adhere to the milestones.

Challenges faced and solutions:

Restricting the scope:

Given the domain as procurement and expense management, we had challenges in restricting the scope of the application. After brainstorming, we incorporated a few scope restrictions on the user types, user roles, approval structure, mapping between purchase requests and purchase orders and the number of suppliers fulfilling an order. Working through our restrictions we made sure that we still

incorporate all the important functionalities of an expense and procurement management system.

UML design:

During the UML design, identifying all the entities, attributes and making sure all dependencies were listed out was quite a challenge. Several reviews with professor followed by discussions and updates, helped immensely to gain better insights into the entire process and also gave a clear picture of the next steps to be taken.

Consolidation of individual work:

As a team, we did a good job of splitting the tasks among ourselves and it ensured that we could move forward in the right pace. But, each of our individual efforts in writing queries necessitated revisiting the UML or the physical schema. Group collaboration tools like Google docs, Google sheets and also the automated features of exporting the script from SQL Server came handy and saved a lot of manual work. Also, we planned to execute our individual tasks coherently so as to reduce any kind of rework. Peer reviews on each other's work and keeping everyone on the team updated on any changes proved to be constructive.

Data population:

Populating the data for all tables created was yet another challenge as the entire process flow had to be thought through and every record added, had to be relevant to the business processes. Moreover, all of the reference integrity constraints had to be preserved. There were instances where we had to go back to create necessary data to generate meaningful results for all queries formed and also to map the business metrics queries to the reports generated. This activity involved lot of manual work and forming subgroups within the team helped focus on different aspects of data creation.

Business Metrics:

To create a visual representation for the queries using reports, our team utilized the report generating functionalities of the visualization tool Tableau.

Initially, it was a challenge to learn how to use the tool, but later we enjoyed creating the reports and also realized the value of data visualization.

Improvements identified on retrospection:

We believe that, probably, doing a bottom-up approach of deciding on the variety and volume of data based on the queries and then creating data would prevent lot of rework. We are happy with every other effort that has been put into the project.

Suggestions:

The project work could also include following tasks:

- Performance tuning of queries to make them more efficient
- Form a buddy team comprising of representatives from every team to address issues at regular intervals and help each other to make sure all the teams stick to the timeline
- Include peer review points to make sure no single student is overloaded with all the work, not every team can be like Expensify :)