PUNJAB UNIVERSITY COLLEGE OF INFORMATION TECHNOLOGY

Course: Database Systems

Submitted to: Dr. Asif Sohail

Submitted by:

- ABDULLAH JAMSHAID (BITF24M031)
- ABDUL HADI (BITF24M005)
- HANZALA AKRAM (BITF24M006)

Work: Project Report

Dated: 22 December, 2025

Database Systems – Project Report

1. Introduction:

The **Pharmacy Management System** is a database-driven application designed to streamline the operations of a modern pharmacy. The system replaces manual record-keeping with an automated relational database that ensures data integrity and efficient retrieval.

The workflow is divided into two distinct roles, managed by separate entities in the database:

1. **Administrative Role (Admin):** The Admin is responsible for Inventory Management. An Admin logs into the system using credentials stored in the **ADMIN** table. Their primary responsibility is to add and manage medicines. The system tracks exactly which Admin added a specific medicine to the inventory, establishing a clear line of accountability (1-to-Many relationship between Admin and Medicines).
2. **Operational Role (Pharmacist):** The Pharmacist is responsible for the Point of Sales (POS). A Pharmacist logs in using credentials from the **PHARMACIST** table. Their main task is to generate sales invoices for customers. The system records every transaction, linking each Invoice to the specific Pharmacist who generated it (1-to-Many relationship between Pharmacist and Invoices).

Key Features:

- **Inventory Tracking:** Medicines are categorized (e.g., Tablet, Syrup) and linked to suppliers via a bridge table to handle multi-supplier scenarios.
- **Sales Processing:** Invoices store transaction headers, while a detailed breakdown of sold items is maintained separately to normalize data.
- **Role Separation:** Security is enforced by physically separating Admin and Pharmacist data into different tables, preventing role overlap.

2. Entity Relationship Diagram (ERD)

The system consists of **8 Entities** connected via relational links. The design focuses on normalization and resolving Many-to-Many relationships using bridge entities.

Entities:

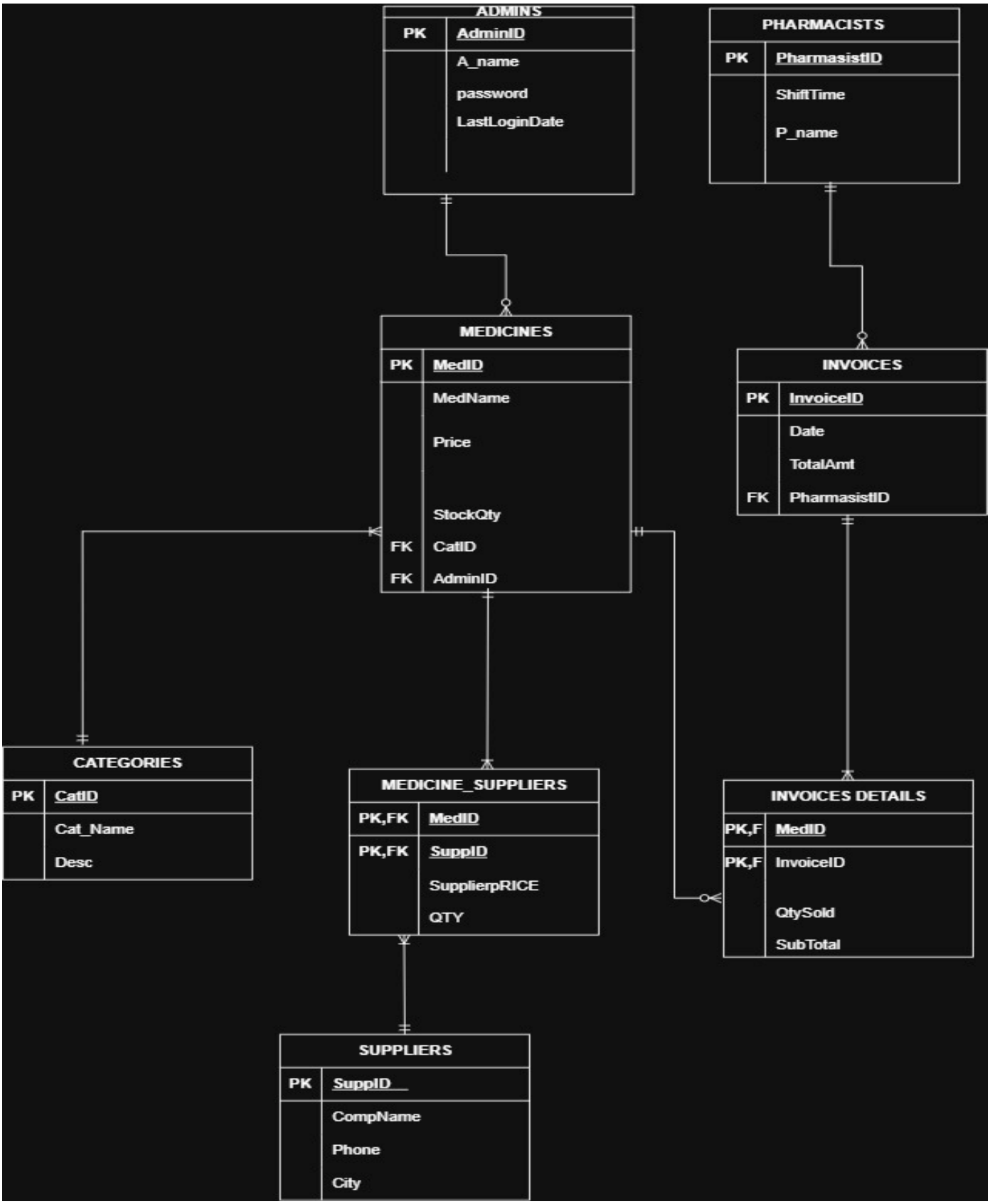
1. **ADMIN:** Stores administrator credentials and login history.
2. **PHARMACIST:** Stores staff details and shift timings.
3. **MEDICINES:** Stores inventory data, linked to the Admin who added it and its Category.
4. **CATEGORIES:** A lookup entity for medicine types (Tablet, Injection, etc.).

5. **SUPPLIERS:** Stores contact information for medicine distributors.
6. **MEDICINE_SUPPLIERS:** A bridge entity resolving the M:N relationship between Medicines and Suppliers (includes Supply Price).
7. **INVOICES:** Stores sales header information (Date, Total), linked to the Pharmacist.
8. **INVOICE_DETAILS:** A bridge entity resolving the M:N relationship between Invoices and Medicines (includes Quantity Sold).

Relationships:

- **Admin — Medicines:** One Admin adds Many Medicines (1:M).
- **Category — Medicines:** One Category classifies Many Medicines (1:M).
- **Pharmacist — Invoices:** One Pharmacist generates Many Invoices (1:M).
- **Invoices — Invoice_Details:** One Invoice contains Many Details (1:M).
- **Medicines — Invoice_Details:** One Medicine appears in Many Invoice Details (1:M).
- **Medicines — Medicine_Suppliers:** One Medicine has Many Suppliers (1:M).
- **Suppliers — Medicine_Suppliers:** One Supplier provides Many Medicines (1:M).

Visual ERD Representation:



3.Construction of the Relational Schema (Normalization)

Unnormalized Form (UNF)

PHARMACY_DATA { InvoiceID, Date, TotalAmt, PharmacistID, P_Name, ShiftTime, MedID, MedName, Price, StockQty, CatID, Cat_Name, Desc, AddedBy_ID, AdminID, A_Name, Password, LastLogin, SupplierID, CompName, Phone, City, SupplyPrice, QTY, QtySold, SubTotal }

First Normal Form (1NF)

Rule Applied: Eliminated repeating groups. All attributes are atomic. Note that Invoice and Supplier data is flattened, causing redundancy (e.g., Invoice Date repeats for every medicine).

MEDICINES_FULL { MedID, MedName, Price, StockQty, AddedBy_ID, AdminID, A_Name, Password, LastLogin, CatID, Cat_Name, Desc }

SUPPLIER_MEDICINES { MedID, SupplierID, CompName, Phone, City, SupplyPrice, QTY }

INVOICE_TRANSACTIONS { InvoiceID, MedID, Date, TotalAmt, GenBY, PharmacistID, P_Name, ShiftTime, QtySold, SubTotal }

Second Normal Form (2NF)

Rule Applied: Eliminated Partial Dependencies. Non-key attributes must depend on the whole Primary Key. We separate the tables where attributes depend only on part of the composite key.

MEDICINES_FULL { MedID, MedName, Price, StockQty, AddedBy_ID, AdminID, A_Name, Password, LastLogin, CatID, Cat_Name, Desc }

SUPPLIERS { SupplierID, CompName, Phone, City }

MEDICINE_SUPPLIERS { MedID, SupplierID, SupplyPrice, QTY } **INVOICES** { InvoiceID, TotalAmt, Date, GenBY, PharmacistID, P_Name, ShiftTime } **INVOICE_DETAILS** { InvoiceID, MedID, QtySold, SubTotal }

Third Normal Form (3NF) & BCNF

Rule Applied: Eliminated Transitive Dependencies. Attributes must depend only on the Primary Key, not on other non-key attributes (e.g., Pharmacist Name depends on PharmacistID, not InvoiceID).

MEDICINES { MedID, Price, MedName, StockQty, AddedBy_ID (FK), CatID (FK) }

CATEGORIES { CatID, Cat_Name, Desc }

ADMIN { AdminID, A_Name, Password, LastLogin } **SUPPLIERS** { SupplierID, CompName, Phone, City }

MEDICINE_SUPPLIERS { MedID (FK), SupplierID (FK), SupplyPrice, QTY }

INVOICES { InvoiceID, TotalAmt, Date, GenBY (FK) }

PHARMACIST { PharmacistID, P_Name, ShiftTime }

INVOICE_DETAILS { InvoiceID (FK), MedID (FK), QtySold, SubTotal }

4.PRIMARY KEYS AND FOREIGN KEYS

1. ADMIN

Primary Key:
AdminID

Foreign Keys:
None

2. PHARMACIST

Primary Key:
PharmacistID

Foreign Keys:
None

3. CATEGORIES

Primary Key:
CatID

Foreign Keys:
None

4. MEDICINES

Primary Key:
MedID

Foreign Keys:
AdminID → ADMIN(AdminID)
CatID → CATEGORIES(CatID)

5. SUPPLIERS

Primary Key:
SupplierID

Foreign Keys:
None

6. MEDICINE_SUPPLIERS

Primary Key:
Composite Key
MedID
SupplierID

Foreign Keys:
MedID → MEDICINES(MedID)
SupplierID → SUPPLIERS(SupplierID)

7. INVOICES

Primary Key:
InvoiceID

Foreign Keys:
GenBY → PHARMACIST(PharmacistID)

8. INVOICE_DETAILS

Primary Key:
Composite Key
InvoiceID
MedID

Foreign Keys:
InvoiceID → INVOICES(InvoiceID)
MedID → MEDICINES(MedID)

5.Top Down Approach

1. Requirement Analysis and Data Collection

For the successful development of the **Pharmacy Management System**, a comprehensive requirement analysis and data collection process is carried out. This involves gathering requirements from all stakeholders including administrators pharmacists and suppliers. The main focus is to understand how medicines are added, managed supplied and sold within the pharmacy. Key data includes administrator login details pharmacist shift information medicine inventory data medicine categories supplier contact details invoice records and sales transactions. Existing issues such as data redundancy manual record keeping and difficulty in tracking medicine sales and suppliers are identified. This analysis ensures that the system supports accurate inventory control efficient billing secure access and reliable data management.

2. Conceptual Design (ER Model Creation)

Based on the collected requirements a conceptual **Entity Relationship Diagram (ERD)** is designed. The ERD identifies the main entities such as Admin Pharmacist Medicines Categories Suppliers Invoices and bridge entities like Medicine_Suppliers and Invoice_Details. Relationships between entities are defined to represent real world interactions such as one admin adding multiple medicines one pharmacist generating many invoices and many to many relationships between medicines and suppliers as well as invoices and medicines.

3. Conversion of ERD Model to Relational Model

The ER model is converted into a relational model by transforming each entity into a table. One to many relationships are handled using foreign keys while many to many relationships are resolved using bridge tables. As a result relational tables such as MEDICINES SUPPLIERS INVOICES MEDICINE_SUPPLIERS and INVOICE_DETAILS are formed. This conversion ensures data consistency and supports relational database implementation.

4. Identification of Primary and Foreign Keys

Based on the ERD and the relational tables formed the following keys are identified.

Primary Keys (PK)

• ADMIN: AdminID
• PHARMACIST: PharmacistID
• CATEGORIES: CatID
• MEDICINES: MedID
• SUPPLIERS: SupplierID
• MEDICINE_SUPPLIERS: Composite Key (MedID SupplierID)
• INVOICES: InvoiceID
• INVOICE_DETAILS: Composite Key (InvoiceID MedID)

Foreign Keys (FK)

• MEDICINES:
o AdminID → ADMIN(AdminID)
o CatID → CATEGORIES(CatID)

• MEDICINE_SUPPLIERS:
o MedID → MEDICINES(MedID)
o SupplierID → SUPPLIERS(SupplierID)

• INVOICES:
o GenBY → PHARMACIST(PharmacistID)

• INVOICE_DETAILS:
o InvoiceID → INVOICES(InvoiceID)
o MedID → MEDICINES(MedID)

5. Refinement and Validation of the Schema

The database schema is refined and validated through the process of normalization. Functional dependencies are analyzed and redundancy is removed by converting the schema into Third Normal Form and BCNF. Transitive and partial dependencies are eliminated and all non key attributes depend only on the primary key. This ensures data integrity reduces anomalies and results in a well structured and efficient database design.

6. Description of the relations:

Table:

ADMIN

Attribute	Data Type	Size	Constraints
AdminID	Varchar2	10	Primary Key
A_Name	Varchar2	30	Not Null
Password	Varchar2	20	Not Null
LastLogin	Date	-	Null allowed

Table:

PHARMACIST

Attribute	Data Type	Size	Constraints
PharmacistID	Varchar2	10	Primary Key
P_Name	Varchar2	30	Not Null
ShiftTime	Varchar2	20	Not Null

Table:

CATEGORIES

Attribute	Data Type	Size	Constraints
CatID	Varchar2	10	Primary Key
Cat_Name	Varchar2	20	Not Null
Desc	Varchar2	50	Null allowed

Table:

MEDICINES

Attribute	Data Type	Size	Constraints
MedID	Varchar2	10	Primary Key
MedName	Varchar2	30	Not Null
Price	Number	-	Not Null, Check(price>=0)
StockQty	Number	-	Not Null, Check(StockQty>=0)
AddedBy_ID	Varchar2	10	Foreign Key → ADMIN(AdminID)
CatID	Varchar2	10	Foreign Key →CATEGORIES(CatID)

Table:

SUPPLIERS

Attribute	Data Type	Size	Constraints
SupplierID	Varchar2	10	Primary Key
CompName	Varchar2	30	Not Null
Phone	Varchar2	15	Not Null
City	Varchar2	20	Not Null

Table:

MEDICINE_SUPPLIERS

Attribute	Data Type	Size	Constraints
MedID	Varchar2	10	Foreign Key → MEDICINES(MedID)
SupplierID	Varchar2	10	Foreign Key → SUPPLIERS(SupplierID)
SupplyPrice	Number	-	Not Null
QTY	Number	-	Not Null
Primary Key	-	-	(MedID, SupplierID) Composite

Table:

INVOICES

Attribute	Data Type	Size	Constraints
InvoiceID	Varchar2	10	Primary Key
TotalAmt	Number	-	Not Null
Date	Date	-	Not Null
GenBY	Varchar2	10	Foreign Key → PHARMACIST(PharmacistID)

Table:

INVOICE_DETAILS

Attribute	Data Type	Size	Constraints
InvoiceID	Varchar2	10	Foreign Key → INVOICES(InvoiceID)
MedID	Varchar2	10	Foreign Key → MEDICINES(MedID)
QtySold	Number	-	Not Null
SubTotal	Number	-	Not Null
Primary Key	-	-	(InvoiceID, MedID) Composite

CREATE TABLE statements:

```
CREATE TABLE ADMIN (
  AdminID     NUMBER(5),
  A_Name      VARCHAR2(50) NOT NULL UNIQUE,
  Password    VARCHAR2(50) NOT NULL,
  LastLogin   DATE,
  CONSTRAINT pk_admin PRIMARY KEY (AdminID)
);

CREATE TABLE PHARMACIST (
  PharmacistID  NUMBER(5),
  P_Name      VARCHAR2(50) NOT NULL UNIQUE,
  ShiftTime   VARCHAR2(20),
  CONSTRAINT pk_pharmacist PRIMARY KEY (PharmacistID)
);

CREATE TABLE CATEGORIES (
  CatID       NUMBER(3),
  Cat_Name    VARCHAR2(50) NOT NULL UNIQUE,
  Description  VARCHAR2(200),
  CONSTRAINT pk_categories PRIMARY KEY (CatID)
);
```

```
CREATE TABLE SUPPLIERS (
  SupplierID   NUMBER(5),
  CompName     VARCHAR2(100) NOT NULL,
  Phone        VARCHAR2(15) NOT NULL,
  City         VARCHAR2(50) NOT NULL,
  CONSTRAINT pk_suppliers PRIMARY KEY (SupplierID)
);

CREATE TABLE MEDICINES (
  MedID        NUMBER(10),
  MedName      VARCHAR2(100) NOT NULL,
  Price        NUMBER(10, 2) Check(price>=0),
  StockQty     NUMBER(5) Check(StockQty>=0) DEFAULT 0,
  AddedBy_ID   NUMBER(5),
  CatID        NUMBER(3),
  CONSTRAINT pk_medicines PRIMARY KEY (MedID),
  CONSTRAINT fk_med_admin FOREIGN KEY (AddedBy_ID) REFERENCES
ADMIN(AdminID),
  CONSTRAINT fk_med_cat FOREIGN KEY (CatID) REFERENCES CATEGORIES(CatID)
);

CREATE TABLE MEDICINE_SUPPLIERS (
  MedID        NUMBER(10),
  SupplierID   NUMBER(5),
  SupplyPrice  NUMBER(10, 2),
  QTY          NUMBER(5),
```

```
  CONSTRAINT pk_med_supp PRIMARY KEY (MedID, SupplierID),
  CONSTRAINT fk_ms_med FOREIGN KEY (MedID) REFERENCES MEDICINES(MedID),
  CONSTRAINT fk_ms_supp FOREIGN KEY (SupplierID) REFERENCES
SUPPLIERS(SupplierID)
);

CREATE TABLE INVOICES (
  InvoiceID    NUMBER(10),
  TotalAmt     NUMBER(12, 2) DEFAULT 0,
  InvDate      DATE DEFAULT SYSDATE,
  GenBY        NUMBER(5),
  CONSTRAINT pk_invoices PRIMARY KEY (InvoiceID),
  CONSTRAINT fk_inv_pharm FOREIGN KEY (GenBY) REFERENCES
PHARMACIST(PharmacistID)
);

CREATE TABLE INVOICE_DETAILS (
  DetailID     NUMBER(10),
  InvoiceID    NUMBER(10),
  MedID        NUMBER(10),
  QtySold      NUMBER(4),
  SubTotal     NUMBER(10, 2),
  CONSTRAINT pk_inv_details PRIMARY KEY (DetailID),
  CONSTRAINT fk_id_inv FOREIGN KEY (InvoiceID) REFERENCES INVOICES(InvoiceID),
  CONSTRAINT fk_id_med FOREIGN KEY (MedID) REFERENCES MEDICINES(MedID)
);
```

7.Queries Statement:

```
SELECT *
FROM ADMIN;
```

Results Explain Describe Saved SQL History

ADMINID	A_NAME	PASSWORD	LASTLOGIN
1	admin	123	-

1 rows returned in 0.00 seconds Download

```
SELECT *
FROM PHARMACIST;
```

Results Explain Describe Saved SQL History

PHARMACISTID	P_NAME	SHIFTTIME
1	ali	Morning
2	sara	Evening

2 rows returned in 0.01 seconds Download

```
SELECT *
FROM CATEGORIES;
```

Results | Explain | Describe | Saved SQL | Histor

CATID	CAT_NAME	DESCRIPTION
1	Tablet	Solid pills
2	Syrup	Liquid

2 rows returned in 0.01 seconds Download

```
SELECT *
FROM SUPPLIERS;
```

Results | Explain | Describe | Saved SQL | History

SUPPLIERID	COMPNAME	PHONE	CITY
1	Pfizer	-	Lahore
2	GSK	-	Karachi

2 rows returned in 0.00 seconds Download

```
SELECT *
 FROM MEDICINES;
```

Results Explain Describe Saved SQL History

MEDID	MEDNAME	PRICE	STOCKQTY	ADDEDBY_ID	CATID
1	Panadol	10	100	1	1
2	Brufen	20	50	1	2
3	CoughSyrup	120	30	1	2

3 rows returned in 0.01 seconds Download

```
SELECT *
 FROM INVOICES;
```

Results Explain Describe Saved SQL History

INVOICEID	TOTALAMT	INVDATE	GENBY
1	30	12/13/2025	1
2	120	12/13/2025	2

2 rows returned in 0.01 seconds Download

```
SELECT *
 FROM INVOICE_DETAILS;
```

Results | Explain | Describe | Saved SQL | History

DETAILID	INVOICEID	MEDID	QTYSOLD	SUBTOTAL
1	1	1	1	10
2	1	2	1	20
3	2	3	1	120

3 rows returned in 0.01 seconds Download

8.INSERT QUERY

```
INSERT INTO PHARMACIST (PharmacistID, P_Name, ShiftTime)
VALUES (3, 'Ahmed', 'Night');
```

Results Explain Describe Saved SQL History

```
1 row(s) inserted.
```

0.00 seconds

Results Explain Describe Saved SQL History

PHARMACISTID	P_NAME	SHIFTTIME
1	ali	Morning
2	sara	Evening
3	Ahmed	Night

3 rows returned in 0.00 seconds Download

9.UPDATE QUERY

```
UPDATE MEDICINES
SET Price = 15
WHERE MedName = 'Panadol';
```

Results Explain Describe Saved SQL

```
1 row(s) updated.
```

Results Explain Describe Saved SQL Histo

MEDNAME	PRICE	CAT_NAME
Panadol	15	Tablet
Brufen	20	Syrup
CoughSyrup	120	Syrup

3 rows returned in 0.01 seconds Download

10.JOIN QUERIES

Question: Write a query to retrieve the list of medicines along with their prices and respective category names

```
SELECT m.MedName, m.Price, c.Cat_Name
FROM MEDICINES m
JOIN CATEGORIES c ON m.CatID = c.CatID;
```

Results Explain Describe Saved SQL History

MEDNAME	PRICE	CAT_NAME
Panadol	10	Tablet
Brufen	20	Syrup
CoughSyrup	120	Syrup

3 rows returned in 0.02 seconds Download

Question: Write a query to display the details of all invoices, including the Invoice ID, Date, and Total Amount, along with the name of the Pharmacist who generated each invoice.

```
SELECT i.InvoiceID, i.InvDate, i.TotalAmt, p.P_Name
FROM INVOICES i
JOIN PHARMACIST p ON i.GenBY = p.PharmacistID;
```

Results Explain Describe Saved SQL History

INVOICEID	INVDATE	TOTALAMT	P_NAME
1	12/13/2025	30	ali
2	12/13/2025	120	sara

2 rows returned in 0.02 seconds Download

Question: Write a query to display a detailed sales report that includes the Invoice ID, Medicine Name, Quantity Sold, and Subtotal for each item sold.

```
SELECT i.InvoiceID, m.MedName, d.QtySold, d.SubTotal
FROM INVOICE_DETAILS d
JOIN INVOICES i ON d.InvoiceID = i.InvoiceID
JOIN MEDICINES m ON d.MedID = m.MedID;
```

Results Explain Describe Saved SQL History

INVOICEID	MEDNAME	QTYSOLD	SUBTOTAL
1	Panadol	1	10
1	Brufen	1	20
2	CoughSyrup	1	120

3 rows returned in 0.05 seconds Download

Question: Write a master query to display complete transaction details, including Invoice ID, Sale Date, the Pharmacist who sold the item, Medicine Name, Category, Quantity Sold, Subtotal, and the name of the Admin who added the medicine to the stock.

```
SELECT
    i.InvoiceID,
    i.InvDate AS Sale_Date,
    p.P_Name AS Sold_By,
    m.MedName AS Medicine,
    c.Cat_Name AS Category,
    d.QtySold,
    d.SubTotal,
    a.A_Name AS Added_By_Admin
FROM INVOICE_DETAILS d
JOIN INVOICES i ON d.InvoiceID = i.InvoiceID
JOIN PHARMACIST p ON i.GenBY = p.PharmacistID
JOIN MEDICINES m ON d.MedID = m.MedID
JOIN CATEGORIES c ON m.CatID = c.CatID
JOIN ADMIN a ON m.AddedBy_ID = a.AdminID;
```

Results Explain Describe Saved SQL History

INVOICEID	SALE_DATE	SOLD_BY	MEDICINE	CATEGORY	QTYSOLD	SUBTOTAL	ADDED_BY_ADMIN
1	12/13/2025	ali	Panadol	Tablet	1	10	admin
1	12/13/2025	ali	Brufen	Syrup	1	20	admin
2	12/13/2025	sara	CoughSyrup	Syrup	1	120	admin

3 rows returned in 0.03 seconds Download

11.CREATE VIEWS AND QURIES:

1. The "Supplier Pricing & Inventory" View

Description/Question: The inventory manager needs a single list showing every medicine, who supplies it, the cost price (supply price) versus the selling price, and the total quantity available from that specific supplier interaction. **Schema Logic:** Join MEDICINES, SUPPLIERS, and MEDICINE_SUPPLIERS.

```
CREATE OR REPLACE VIEW View_Supplier_Pricing AS
SELECT
  m.MedName,
  s.CompName AS Supplier,
  s.City,
  ms.SupplyPrice AS Cost_Price,
  m.Price AS Retail_Price,
  (m.Price - ms.SupplyPrice) AS Est_Margin_Per_Unit,
  ms.QTY AS Supplier_Provided_Qty
FROM MEDICINES m
JOIN MEDICINE_SUPPLIERS ms ON m.MedID = ms.MedID
JOIN SUPPLIERS s ON ms.SupplierID = s.SupplierID;
```

2. The "Full Sales Audit" View

Description/Question: We need a detailed audit trail that shows every single line item sold, which invoice it belongs to, who sold it (Pharmacist), and the category of the medicine. This helps in tracking specific product movement. **Schema Logic:** Join INVOICES, INVOICE_DETAILS, MEDICINES, CATEGORIES, and PHARMACIST.

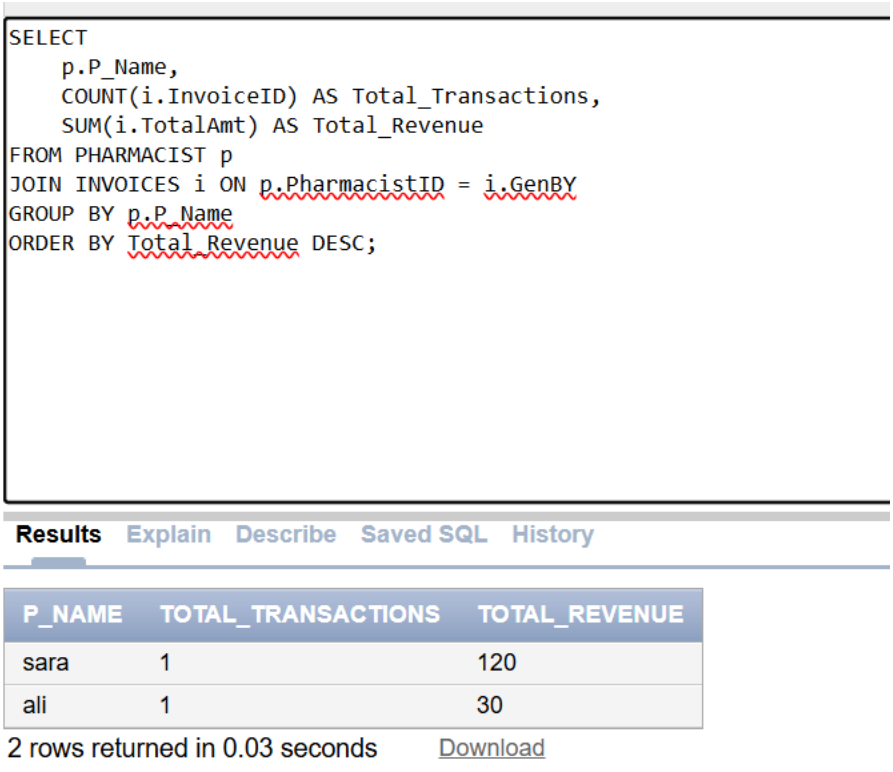
```
CREATE OR REPLACE VIEW View_Full_Sales_Audit AS
SELECT
   i.InvoiceID,
   i.InvDate,
   p.P_Name AS Pharmacist,
   m.MedName,
   c.Cat_Name AS Category,
   d.QtySold,
   d.SubTotal
FROM INVOICE_DETAILS d
JOIN INVOICES i ON d.InvoiceID = i.InvoiceID
JOIN MEDICINES m ON d.MedID = m.MedID
JOIN CATEGORIES c ON m.CatID = c.CatID
JOIN PHARMACIST p ON i.GenBY = p.PharmacistID;
```

Which Pharmacist has generated the highest revenue?

Reasoning: This is essential for performance reviews. We need to sum the TotalAmt from the INVOICES table grouped by the Pharmacist.

SQL

```
SELECT
   p.P_Name,
   COUNT(i.InvoiceID) AS Total_Transactions,
   SUM(i.TotalAmt) AS Total_Revenue
FROM PHARMACIST p
JOIN INVOICES i ON p.PharmacistID = i.GenBY
GROUP BY p.P_Name
ORDER BY Total_Revenue DESC;
```



Results Explain Describe Saved SQL History

P_NAME	TOTAL_TRANSACTIONS	TOTAL_REVENUE
sara	1	120
ali	1	30

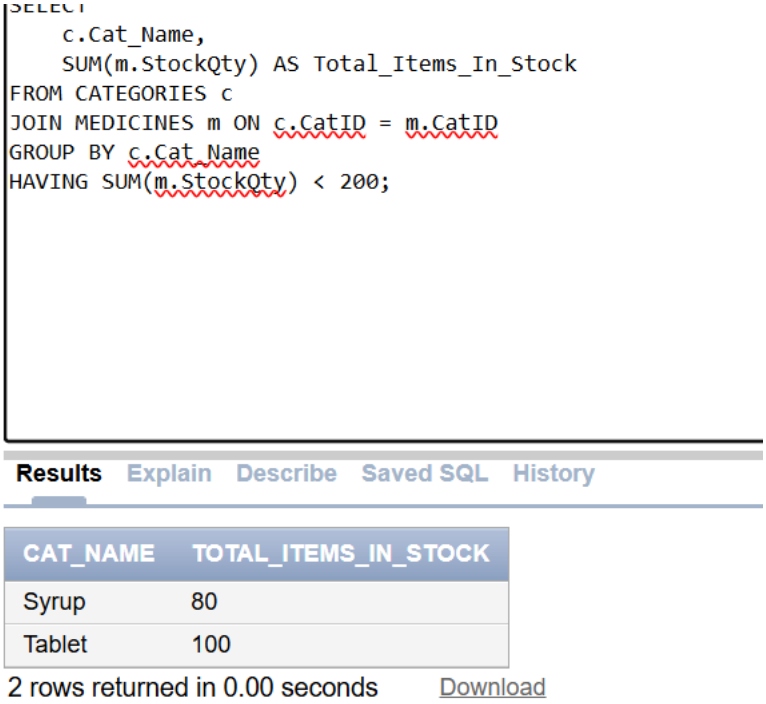
2 rows returned in 0.03 seconds Download

Which Medicine Categories are low in stock?

Reasoning: Instead of just looking at individual medicines, the manager wants to see the total stock available per *Category* (e.g., total Tablets vs. total Syrups) to make bulk purchasing decisions.

SQL

```
SELECT
  c.Cat_Name,
  SUM(m.StockQty) AS Total_Items_In_Stock
FROM CATEGORIES c
JOIN MEDICINES m ON c.CatID = m.CatID
GROUP BY c.Cat_Name
HAVING SUM(m.StockQty) < 200; -- Example threshold
```



```
SELECT
    c.Cat_Name,
    SUM(m.StockQty) AS Total_Items_In_Stock
FROM CATEGORIES c
JOIN MEDICINES m ON c.CatID = m.CatID
GROUP BY c.Cat_Name
HAVING SUM(m.StockQty) < 200;
```

Results Explain Describe Saved SQL History

CAT_NAME	TOTAL_ITEMS_IN_STOCK
Syrup	80
Tablet	100

2 rows returned in 0.00 seconds Download

What is the potential profit currently sitting in stock?

Reasoning: The owner wants to know the financial value of the current inventory. This is calculated by multiplying StockQty by Price for all items in the MEDICINES table.

SQL

```
SELECT
  MedName,
  StockQty,
  Price,
  (StockQty * Price) AS Potential_Revenue_Value
FROM MEDICINES
ORDER BY Potential_Revenue_Value DESC;
```

```
SELECT
    MedName,
    StockQty,
    Price,
    (StockQty * Price) AS Potential_Revenue_Value
FROM MEDICINES
ORDER BY Potential_Revenue_Value DESC;
```

Results Explain Describe Saved SQL History

MEDNAME	STOCKQTY	PRICE	POTENTIAL_REVENUE_VALUE
CoughSyrup	30	120	3600
Panadol	100	15	1500
Brufen	50	20	1000

3 rows returned in 0.00 seconds Download

Query: High-Value Transactions

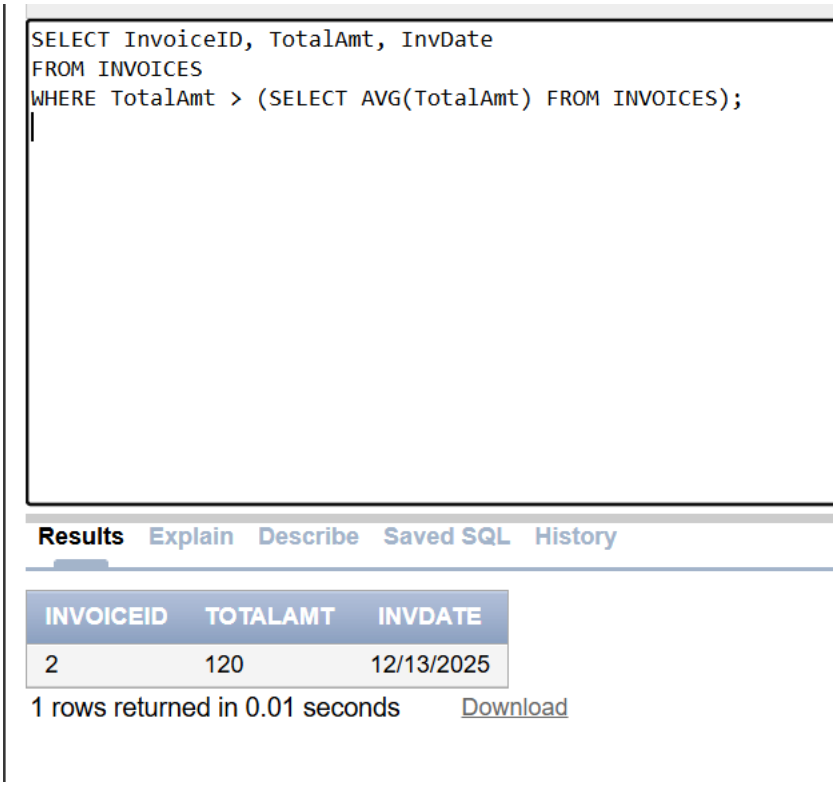
Question: Find all invoices where the total amount is higher than the average invoice amount. **Logic:** Use a subquery to calculate the AVG(TotalAmt) first, then filter INVOICES against it.

SQL

SELECT InvoiceID, TotalAmt, InvDate

FROM INVOICES

WHERE TotalAmt > (SELECT AVG(TotalAmt) FROM INVOICES);



Query: Supplier Concentration by City

Question: In which cities are our suppliers located, and how many are in each? This helps in logistics planning. **Logic:** Group SUPPLIERS by City and count them.

SQL

SELECT City, COUNT(SupplierID) AS Number_Of_Suppliers

FROM SUPPLIERS

GROUP BY City

ORDER BY Number_Of_Suppliers DESC;

```
SELECT City, COUNT(SupplierID) AS Number_Of_Suppliers
FROM SUPPLIERS
GROUP BY City
ORDER BY Number_Of_Suppliers DESC;
```

Results Explain Describe Saved SQL History

CITY	NUMBER_OF_SUPPLIERS
Karachi	1
Lahore	1

2 rows returned in 0.00 seconds Download

Query: Best Selling Medicine

Question: What is our #1 product by quantity sold? **Logic:** Sum QtySold in INVOICE_DETAILS and join with MEDICINES to get the name.

SQL

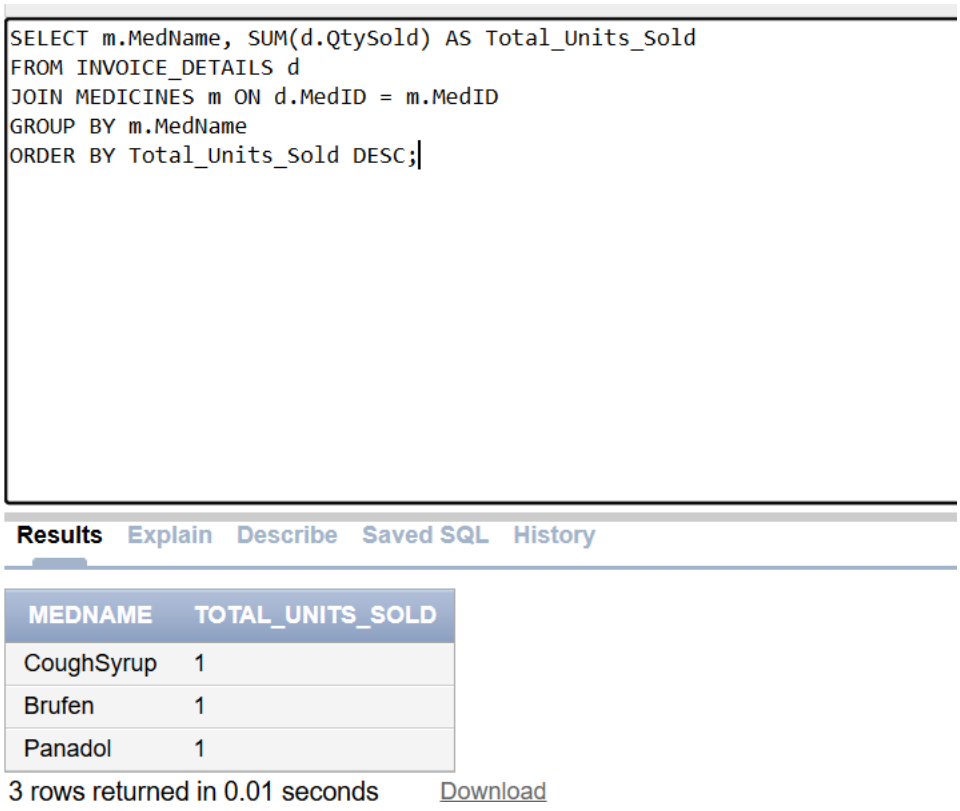
SELECT m.MedName, SUM(d.QtySold) AS Total_Units_Sold

FROM INVOICE_DETAILS d

JOIN MEDICINES m ON d.MedID = m.MedID

GROUP BY m.MedName

ORDER BY Total_Units_Sold DESC;



```
SELECT m.MedName, SUM(d.QtySold) AS Total_Units_Sold
FROM INVOICE_DETAILS d
JOIN MEDICINES m ON d.MedID = m.MedID
GROUP BY m.MedName
ORDER BY Total_Units_Sold DESC;
```

Results Explain Describe Saved SQL History

MEDNAME	TOTAL_UNITS_SOLD
CoughSyrup	1
Brufen	1
Panadol	1

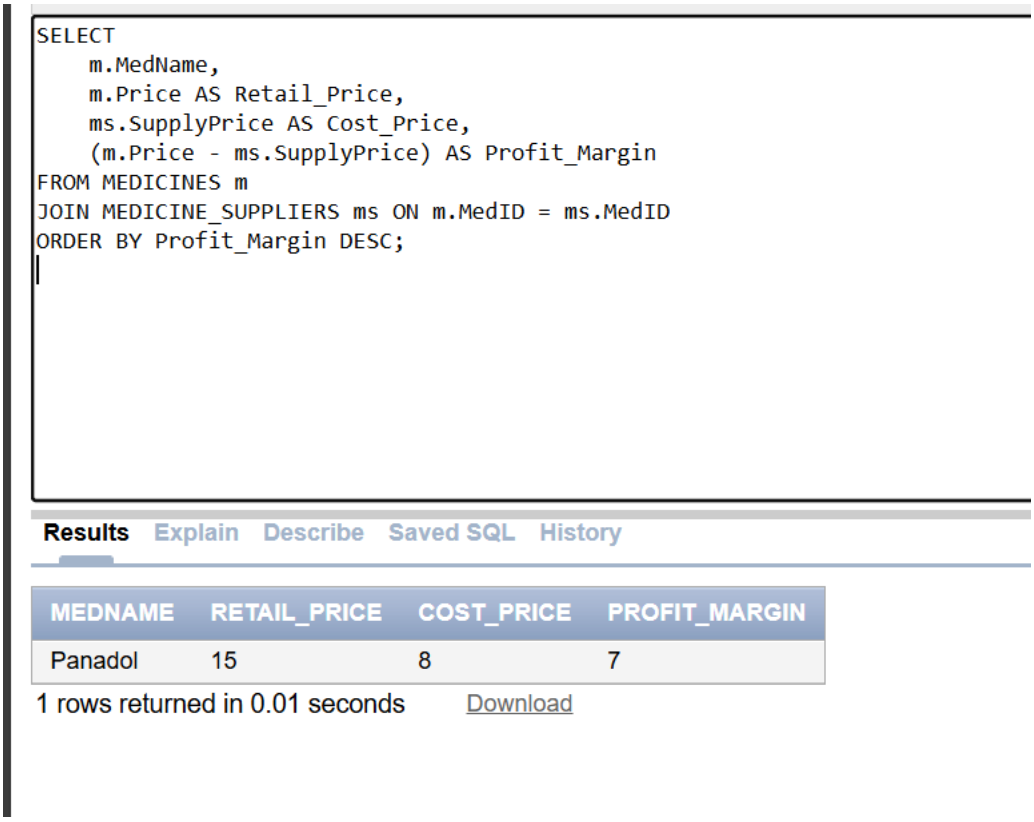
3 rows returned in 0.01 seconds Download

Query: Calculate Profit Margin

Question: For every medicine, what is the profit margin (Retail Price - Supplier Price)?
Logic: Subtract SupplyPrice in MEDICINE_SUPPLIERS from Price in MEDICINES.

SQL

```
SELECT
   m.MedName,
   m.Price AS Retail_Price,
   ms.SupplyPrice AS Cost_Price,
   (m.Price - ms.SupplyPrice) AS Profit_Margin
FROM MEDICINES m
JOIN MEDICINE_SUPPLIERS ms ON m.MedID = ms.MedID
ORDER BY Profit_Margin DESC;
```



1. TRIGGER

Purpose: Reduce medicine stock after a sale is recorded in INVOICE_DETAILS

```
CREATE OR REPLACE TRIGGER trg_update_stock
AFTER INSERT ON INVOICE_DETAILS
FOR EACH ROW
BEGIN
 UPDATE MEDICINES
 SET StockQty = StockQty - :NEW.QtySold
 WHERE MedID = :NEW.MedID;
END;
```

2. PROCEDURE

Purpose: Add a new medicine into MEDICINES table

```
CREATE OR REPLACE PROCEDURE add_medicine
(
 p_medid IN MEDICINES.MedID%TYPE,
 p_name IN MEDICINES.MedName%TYPE,
 p_price IN MEDICINES.Price%TYPE,
 p_stock IN MEDICINES.StockQty%TYPE,
 p_admin IN MEDICINES.AddedBy_ID%TYPE,
 p_cat IN MEDICINES.CatID%TYPE
)
BEGIN
 INSERT INTO MEDICINES
 VALUES (p_medid p_name p_price p_stock p_admin p_cat);
```

```
END;
```

3. FUNCTION

Purpose: Calculate invoice total amount

```
CREATE OR REPLACE FUNCTION get_invoice_total
(
 p_invoiceid IN INVOICE_DETAILS.InvoiceID%TYPE
)
RETURN NUMBER
IS
 total NUMBER;
BEGIN
 SELECT SUM(SubTotal)
 INTO total
 FROM INVOICE_DETAILS
 WHERE InvoiceID = p_invoiceid;

 RETURN total;
END;
```