TOPICS TO BE COVERED:

* ~~DISTINCT~~
* ~~AND~~, ~~OR~~
* ~~NULL~~
* OT
* ALL JOINS – INNER, ~~LEFT~~, RIGHT, OUTER, EQUI, NATURAL
* ~~UNION~~, INTERSECTION
* ~~IN~~, ANY, SOME, ALL
* ~~SUM~~, ~~COUNT~~, MIN, MAX, ~~AVG~~, MEDIAN
* ~~Subquery~~
* ~~EXISTS~~
* ~~HAVING~~
* ~~WITH~~
* ~~PIVOT~~
* ~~LEAD~~, LAG
* ~~RANK~~
* ~~ROLL UP~~
* ~~CONCATENATION~~
* ~~COALESCE~~
* ~~IN LINE SELECT~~
* ~~FETCH FIRST ….~~
* VIEWS – MATERIALIZED, DYNAMIC (use for employee data)
* ~~MINUS~~

1. Which store has highest customer retention (flag for determining purchased online or offline - order table) – **(Minita & TJ)**

WITH returning\_customers AS (

SELECT customerID, COUNT(DISTINCT orderID) AS No\_of\_orders, Quarter, storeid

FROM (SELECT orderid, customerid, storeid,

(CASE WHEN EXTRACT(MONTH from orderdate) BETWEEN 01 AND 03 THEN 'Q1'

WHEN EXTRACT(MONTH from orderdate) BETWEEN 04 AND 06 THEN 'Q2'

WHEN EXTRACT(MONTH from orderdate) BETWEEN 07 AND 09 THEN 'Q3'

ELSE 'Q4' END) as Quarter

FROM orders) order\_quarter

WHERE storeid IS NOT NULL

GROUP BY customerID, Quarter, storeid

HAVING COUNT(orderID) > 1

ORDER BY customerID)

SELECT city, rc.storeID AS Store, COUNT(rc.customerid) AS CustomerCount

FROM returning\_customers RC

JOIN Store S ON RC.storeid = S.storeid

GROUP BY city, rc.storeID

order by CustomerCount desc;

---------------------------------------------------------------------

1. Find the top 3 best-selling online products for each category in 2023, based on their sales and the average order fulfillment time. Order it in descending order according to their sales. Include taxes and discount based on the active promotions on the products.  
   **(Rashmi)**

WITH TopProducts AS (

SELECT

c.c\_name as CategoryName,

p.p\_name as ProductName,

TO\_CHAR(SUM((od.unit\_price \* od.quantity) \* (1 - NVL(pr.discount\_amount / 100, 0)) \* (1 + p.tax)), '$9999999.99') Sales,

count(o.orderid) AS total\_orders,

FLOOR(AVG(delivery\_date - OrderDate)) Avg\_Fulfillment\_Time,

RANK() OVER ( partition by c.c\_name order by SUM((od.unit\_price \* od.quantity) \* (1 - NVL(pr.discount\_amount / 100, 0)) \* (1 + p.tax)) desc, AVG(delivery\_date - OrderDate) asc) rnk

FROM product p

JOIN subcategory sc on p.SUB\_CATEGORY\_ID = sc.SUB\_CATEGORY\_ID

join productcategory c on sc.categoryid = c.categoryID

JOIN order\_details od ON p.productid = od.productid

JOIN orders o ON od.orderid = o.orderid

join promotions pr on p.productid = pr.productid

JOIN Delivery d on d.order\_id = o.orderid

WHERE extract(year from o.orderdate) in ('2023')

and order\_flag = 'Online' and OrderStatus = 'Confirmed'

GROUP BY c.c\_name, p.p\_name

Order by 1,3 desc)

Select \* from TopProducts where rnk <=3;

---------------------------------------------------------------------

1. Identify suppliers across categories in every warehouse who consistently deliver products late for these categories. Every category can have multiple suppliers. Late deliveries are defined as someone who supplies after 10 days of request from the warehouse. In addition, also track the average number of days per category where the stock is stagnant at the warehouse. (Insight - Demand Supply optimization) - Smruti

SELECT w.WarehouseID, pc.c\_name,

ceil(avg(sm.outbound\_date - sm.inbound\_date)) as StockMovement, -- how many days it takes (if this is more, then product is at the warehouse for very long and there is no demand)

(CASE WHEN AVG(W\_Supply\_Date - W\_Request\_Date) > 9 THEN SName END) AS Supplier\_Name -- worst suppliers

FROM Warehouses w

join stock sm on w.warehouseid = sm.warehouseid

Join Product p on sm.productid = p.productid

Join subcategory sc on p.sub\_category\_id = sc.sub\_category\_id

join productcategory pc on pc.categoryid = sc.categoryID

JOIN purchase\_details pd on p.productid = pd.productid

JOIN PURCHASES p on p.Purchaseid = pd.purchaseid

JOIN SUPPLIES sp on p.productID = sp.productID

JOIN SUPPLIERS sup on sp.supplierID = sup.supplierID

GROUP BY w.WarehouseID, pc.c\_name, sname

order by warehouseid,Supplier\_Name desc, StockMovement desc;

---------------------------------------------------------------------

1. Identify the top 5 most valuable online customers in terms of their lifetime spending. For these top customers, determine average delivery fullfillment time, if there are any specific product categories that these customers frequently order. Provide a breakdown of revenue by product category for each of these top customers. (HomeDepot will get to know how to reduce the avg delivery times for these type of customers) - **Minita & TJ**

WITH category\_info AS (

SELECT

o.CustomerID, o.orderid, sc.CategoryID, pc.C\_name,

SUM(od.Quantity \* od.Unit\_Price) AS Category\_Revenue,

DENSE\_RANK() OVER (partition by o.customerid ORDER BY SUM(od.Quantity \* od.Unit\_Price) DESC) AS rank\_category

--DENSE\_RANK() OVER (PARTITION BY o.CustomerID ORDER BY COUNT(o.OrderID) DESC) AS rank\_category

FROM Orders o

JOIN Order\_details od ON od.OrderID = o.OrderID

JOIN product P ON od.productID = P.productID

JOIN SubCategory sc ON P.Sub\_category\_ID = sc.Sub\_category\_ID

JOIN ProductCategory pc ON pc.CategoryID = sc.CategoryID

WHERE UPPER(o.Order\_Flag) = 'ONLINE'

GROUP BY o.orderid, o.CustomerID, sc.CategoryID, pc.C\_name

ORDER BY o.CustomerID

)

SELECT

ci.CustomerID,

SUM(od.Quantity \* od.Unit\_Price) AS order\_total,

round(AVG(d.Delivery\_Date - cast(o.OrderDate as date)),3) AS avg\_delivery\_time,

ci.C\_name,

ci.Category\_Revenue,

concat(round((ci.Category\_Revenue/SUM(od.Quantity \* od.Unit\_Price)) \* 100,2),'%') As Spent\_Percent

FROM category\_info ci

JOIN Order\_details od ON od.OrderID = ci.OrderID

JOIN Delivery d ON d.Order\_ID = od.OrderID

JOIN Orders o ON o.OrderID = od.OrderID

WHERE ci.rank\_category = 1

GROUP BY ci.CustomerID, ci.C\_name, ci.Category\_Revenue

ORDER BY order\_total DESC

FETCH FIRST 5 ROWS WITH TIES;

---------------------------------------------------------------------

`

1. Sentiment analysis depending on keyword - bad or good quality. Make separate tables for positive and negative reviews. - **Minita & TJ**

WITH rev AS (SELECT p.productid,p\_name,

(CASE WHEN UPPER(review\_description) LIKE UPPER('%bad%') OR

UPPER(review\_description) LIKE UPPER('%waste%') OR

UPPER(review\_description) LIKE UPPER('%pathetic%') OR

UPPER(review\_description) LIKE UPPER('%worst%') THEN 'Negative'

WHEN UPPER(review\_description) LIKE UPPER('%good%') OR

UPPER(review\_description) LIKE UPPER('%great%') OR

UPPER(review\_description) LIKE UPPER('%excellent%') OR

UPPER(review\_description) LIKE UPPER('%best%') OR

UPPER(review\_description) LIKE UPPER('%fantastic%') OR

UPPER(review\_description) LIKE UPPER('%amazing%') THEN 'Positive'

END) AS Review\_Type

FROM Reviews r

JOIN product p ON p.productid = r.productid)

SELECT p\_name, P\_count\_rev as Positive\_Reviews, N\_count\_rev as Negative\_Reviews,

(CASE WHEN P\_count\_rev/(P\_count\_rev + N\_count\_rev) >= 0.6 THEN 'Good Quality'

WHEN N\_count\_rev/(P\_count\_rev + N\_count\_rev) >= 0.6 THEN 'Bad Quality'

ELSE 'Neutral' END) as Quality

FROM rev

PIVOT (

COUNT(productid) AS count\_rev

FOR review\_type IN ('Positive' AS P, 'Negative' AS N))

ORDER BY Positive\_Reviews DESC ;

---------------------------------------------------------------------

6. As a store manager of S001, you want to assess the impact of recent promotional campaigns where the discount provided on the products is more than 10% on employee performance across different departments in store.

Identify departments where at least one employee made a successful sale with a promotional discount for the last 6 months.

with store\_combine as (

select \* from form\_cashier union

select \* from form\_managers union

select \* from form\_attendants )

SELECT distinct C\_Name

FROM productcategory pc

WHERE pc.categoryid IN (

SELECT c.categoryid

FROM store\_combine st

JOIN shift\_crew sc on st.shift\_crewid = sc.shift\_crewid

join crew\_store cs on sc.shift\_crewid = cs.shift\_crewid

JOIN Store s on s.storeid = cs.storeid

Join store\_availability sa on s.storeid = sa.storeid

JOIN product p ON sa.productid = p.productid

JOIN subcategory sub on p.SUB\_CATEGORY\_ID = sub.SUB\_CATEGORY\_ID

join productcategory c on sub.categoryid = c.categoryID

JOIN order\_details od ON p.productid = od.productid

JOIN orders o ON od.orderid = o.orderid

join promotions pr on p.productid = pr.productid

WHERE to\_date(orderdate) >= ADD\_MONTHS(SYSDATE, -6)

AND pr.discount\_amount > 10 and status = 'Active'

and s.STOREID = 'S001');

---------------------------------------------------------------------

7. As a Home Depot employee, I want to retrieve customer post-order statistics. The query should cover the following aspects:

* Retrieve the total number of returns and exchanges processed by each customer
* Identify the top product with the highest number of returns and exchanges(to analyze churn) - Aps

WITH ProductSummary AS (

SELECT p.PRODUCTID, p.MODEL\_NO AS "Model Number", p.P\_NAME AS "Product Name", p.BRAND,

COUNT(DISTINCT eo.REQUESTID) AS EXCHANGE\_COUNT,

COUNT(DISTINCT rc.REQUESTID) AS RETURN\_COUNT,

NVL(SUM(r.REFUND\_AMOUNT), 0) AS TOTAL\_REFUND\_AMOUNT

FROM Product p

LEFT JOIN Post\_Order\_Status pos ON p.PRODUCTID = pos.PRODUCTID

LEFT JOIN Exchange\_Orders eo ON pos.REQUESTID = eo.REQUESTID

LEFT JOIN Return\_Cancel rc ON pos.REQUESTID = rc.REQUESTID

LEFT JOIN REFUND r ON rc.REFUNDID = r.REFUNDID

GROUP BY p.PRODUCTID, p.MODEL\_NO, p.P\_NAME, p.BRAND

)

SELECT PRODUCTID, "Model Number", "Product Name", BRAND, EXCHANGE\_COUNT, RETURN\_COUNT, TOTAL\_REFUND\_AMOUNT

FROM ProductSummary

ORDER BY (EXCHANGE\_COUNT + RETURN\_COUNT) DESC

FETCH FIRST 3 ROWS ONLY;

---------------------------------------------------------------------

8. Given query calculates the performance difference percentage of each cashier employee compared to the overall average sales of all cashier employees.

with employeeperformance as (

select

e.f\_name,

e.l\_name,

sum(finalamount) AS TotalSales

FROM

Employees e JOIN cashier c on e.employee\_id = c.employeeid

join payments p on e.employee\_id = p.employeeid

GROUP BY e.f\_name, e.l\_name),

OverallAveragePerformance AS (

SELECT

sum(finalamount) AS OverallSales

FROM

Employees e JOIN cashier c on e.employee\_id = c.employeeid

join payments p on e.employee\_id = p.employeeid

)

SELECT

ep.f\_name AS EmployeeFirstName,

ep.l\_name AS EmployeeLastName,

concat(round(((oap.OverallSales - ep.TotalSales) / oap.OverallSales) \* 100,2), '%') AS PerformanceDifferencePercentage

FROM

EmployeePerformance ep

JOIN

OverallAveragePerformance oap ON 1=1 ;

---------------------------------------------------------------------

9. Given a table named SALES that contains daily sales data, the goal is to determine the percentage increase in daily sales compared to the day before. (Aps)

WITH DailySales AS (

SELECT TO\_DATE(orderdate,'DD-MON-YYYY') as orderdate,

SUM((od.unit\_price \* od.quantity) \* (1 - NVL(pr.discount\_amount / 100, 0)) \* (1 + p.tax)) AS daily\_sales

FROM ORDER\_DETAILS od

JOIN ORDERS o ON od.orderid = o.orderid

LEFT JOIN PROMOTIONS pr ON od.productid = pr.productid

LEFT JOIN PRODUCT p ON od.productid = p.productid

GROUP BY TO\_DATE(orderdate,'DD-MON-YYYY')

),

DailySalesWithPrevDay AS (

SELECT orderdate, daily\_sales,

LAG(daily\_sales) OVER (ORDER BY orderdate) AS prev\_day\_sales

FROM DailySales

)

SELECT orderdate,

TO\_CHAR(daily\_sales, '$99999999.99') AS daily\_sales,

TO\_CHAR(NVL(prev\_day\_sales, '0'), '$99999990.99') AS prev\_day\_sales,

TO\_CHAR(NVL((daily\_sales - prev\_day\_sales) / NVL(prev\_day\_sales, 1) \* 100,'0'),'FM9999999990.00') || '%' AS percentage\_increase

FROM DailySalesWithPrevDay;

---------------------------------------------------------------------

10. The company wants to monitor changes in the quantity of products in stock over time to identify trends and potential issues. Develop a SQL query using the LEAD and LAG functions to compare the current stock quantity with the previous and next periods. **(TJ & Minita)**

SELECT ProductID, Stock\_date, TO\_CHAR(Current\_Quantity) Current\_Quantity,

TO\_CHAR(Previous\_Quantity) Previous\_Quantity,

COALESCE(TO\_CHAR(Former\_Quantity), 'NA') Former\_Quantity,

COALESCE(TO\_CHAR((Current\_Quantity - Previous\_Quantity)), 'NA') AS Difference,

COALESCE(ROUND(((Current\_Quantity - Previous\_Quantity)/Previous\_Quantity\*100),2),0) || '%' AS Percent\_Change

FROM (

SELECT ProductID, Stock\_date, Quantity AS Current\_Quantity,

LAG(Quantity) OVER (PARTITION BY ProductID ORDER BY Stock\_date) AS Previous\_Quantity,

LAG(Quantity, 2) OVER (PARTITION BY ProductID ORDER BY Stock\_date) AS Former\_Quantity

FROM Stock

ORDER BY ProductID, Stock\_date)

WHERE Previous\_Quantity IS NOT NULL

---------------------------------------------------------------------

11. Query to find which products are available online or offline exclusively - (**Minita & TJ)**

WITH store\_prods AS (

SELECT p\_name, SP.productID

FROM store\_product SP

JOIN product P ON SP.productID = P.productID

MINUS

SELECT p\_name, OP.productID

FROM online\_product OP

JOIN product P ON OP.productID = P.productID),

online\_prods AS (

SELECT p\_name, OP.productID

FROM online\_product OP

JOIN product P ON OP.productID = P.productID

MINUS

SELECT p\_name, SP.productID

FROM store\_product SP

JOIN product P ON SP.productID = P.productID)

SELECT 'Store', productID, p\_name FROM store\_prods

UNION

SELECT 'Online', productID, p\_name FROM online\_prods;

---------------------------------------------------------------------

12. Query to calculate and provide insights on the sales performance of different subcategories of products. (Neha)

SELECT SC\_Name AS Subcategory\_Name, SUM(OD.Quantity) AS Total\_Quantity\_Sold,

TO\_CHAR(SUM(OD.Quantity \* OD.Unit\_Price), '$999,999,999.99') AS Total\_Revenue,

CONCAT(ROUND((SUM(OD.Quantity \* OD.Unit\_Price) / (SELECT SUM(OD.Quantity \* OD.Unit\_Price)

FROM ProductCategory PC

JOIN Subcategory SC ON PC.categoryID = SC.categoryID

JOIN Product P ON P.Sub\_category\_ID = SC.Sub\_category\_ID

JOIN Order\_Details OD ON P.PRODUCTID = OD.PRODUCTID

JOIN Orders O ON OD.OrderID = O.OrderID)),4) \* 100, '%') AS Revenue\_Percentage

FROM ProductCategory PC

JOIN Subcategory SC ON PC.categoryID = SC.categoryID

JOIN Product P ON P.Sub\_category\_ID = SC.Sub\_category\_ID

JOIN Order\_Details OD ON P.PRODUCTID = OD.PRODUCTID

JOIN Orders O ON OD.OrderID = O.OrderID

GROUP BY ROLLUP (SC\_Name)

HAVING sc\_name IS NOT NULL

ORDER BY revenue\_percentage desc;

---------------------------------------------------------------------

-- Queries to create additional tables

CREATE TABLE SHIPMENT\_CONSISTS\_OF (

Shipment\_Number VARCHAR2(10) CONSTRAINT shipno\_fk REFERENCES shipment,

ProductID VARCHAR2(50) CONSTRAINT prodid\_fk REFERENCES products);

commit;

CREATE TABLE PRODUCTS\_BELONG\_TO (

ProductID VARCHAR2(50) CONSTRAINT productid\_fk REFERENCES products,

CategoryID VARCHAR2(50) CONSTRAINT catid\_fk REFERENCES subcategory);

commit;

CREATE TABLE DELIVERY (

TrackingNumber VARCHAR2(20) CONSTRAINT trackno\_pk PRIMARY KEY,

Carrier VARCHAR2(20),

Delivery\_Date Date,

Delivery\_Person VARCHAR2 (50),

Shipping\_Address VARCHAR2 (50),

Order\_ID VARCHAR2(20) CONSTRAINT delivery\_fk REFERENCES Orders

);

commit;

CREATE TABLE ORDERS (

OrderID VARCHAR2(20) CONSTRAINT orderid\_pk PRIMARY KEY,

Orderitem VARCHAR2(50)CONSTRAINT orderitem\_nn NOT NULL,

CustomerID VARCHAR2(20) CONSTRAINT custid\_fk REFERENCES Customer,

PaymentID VARCHAR2(20) CONSTRAINT paymentid\_fk REFERENCES Payments,

OrderDate TIMESTAMP,

OrderStatus VARCHAR2(50) CONSTRAINT orderstatus\_nn CHECK (OrderStatus IN ('Placed', 'Pending', 'Canceled'))

);

commit;

CREATE TABLE PAYMENTS(

PaymentID VARCHAR2(20) CONSTRAINT paymentid\_pk PRIMARY KEY,

Paymentdate TIMESTAMP,

Payment\_Method VARCHAR2(20) CONSTRAINT method\_nn CHECK (Payment\_Method IN ('Credit Card', 'Cash', 'PayPal')),

Payment\_Amount NUMBER (10,2) CONSTRAINT amount\_pos CHECK (Payment\_Amount > 0),

Tax NUMBER (4,2),

Discount NUMBER (4,2),

EmployeeID VARCHAR2(50) CONSTRAINT empid\_fk REFERENCES Employees,

Payment\_Status VARCHAR2(50) CONSTRAINT status\_nn CHECK (Payment\_Status IN ('Paid', 'Pending', 'Failed', 'Canceled'))

);

commit;

CREATE TABLE MANAGERS (

EmployeeID VARCHAR2(50) CONSTRAINT managerid\_fk REFERENCES EMPLOYEES,

Workforce\_Managed NUMBER(5)

);

commit;

CREATE TABLE ATTENDANTS (

EmployeeID VARCHAR2(50) CONSTRAINT attendantid\_fk REFERENCES EMPLOYEES,

Security\_Access\_Level VARCHAR2(10) CONSTRAINT security\_check CHECK (Security\_Access\_Level IN ('Online', 'Physical', 'Both'))

);

commit;

CREATE TABLE CASHIER (

EmployeeID VARCHAR2(50) CONSTRAINT cashierid\_fk REFERENCES EMPLOYEES,

Register\_Number NUMBER(3)

);

commit;

CREATE TABLE EMPLOYEE\_ADDRESS (

EmployeeID VARCHAR2(50) CONSTRAINT employeeid\_fk REFERENCES EMPLOYEES, Street VARCHAR2(50),

City VARCHAR2(50),

Apartment\_Number NUMBER(4),

Zipcode NUMBER(4),

PRIMARY KEY (EmployeeID, Street, City, Apartment\_Number, Zipcode)

);

commit;

CREATE TABLE EMPLOYEE\_SUPERVISOR (

SupervisorID VARCHAR2(50) CONSTRAINT supervisorid\_fk REFERENCES EMPLOYEES,

Employees\_Supervised NUMBER (3)

);

commit;

CREATE TABLE INVOICE (

Invoice\_Number VARCHAR2(50) CONSTRAINT invoiceno\_pk PRIMARY KEY,

Invoice\_Date DATE,

PaymentID VARCHAR2(20) CONSTRAINT invpaymentid\_fk REFERENCES Payments,

Billing\_address VARCHAR2(255),

Total\_Amount NUMBER(10,2)

);

commit;

CREATE TABLE ORDER\_DETAILS (

OrderID VARCHAR2(20) CONSTRAINT orderdetails\_fk REFERENCES Orders,

Quantity NUMBER(3),

ProductID VARCHAR2(50) CONSTRAINT orderproductid\_fk REFERENCES Products,

Unit\_Price NUMBER(10, 2)

);

commit;

CREATE TABLE STORE\_AVAILABILITY (

StoreID VARCHAR2(50),

ProductID VARCHAR2(50),

Quantity NUMBER(4),

sadate DATE,

PRIMARY KEY (StoreID, ProductID, sadate),

FOREIGN KEY (Storeid) REFERENCES Store,

FOREIGN KEY (ProductID) REFERENCES Products

);

commit;

CREATE TABLE POST\_ORDER\_STATUS (

RequestID VARCHAR2(50) CONSTRAINT reqid\_pk PRIMARY KEY,

Reason VARCHAR2(255),

ProductID VARCHAR2(50) CONSTRAINT postorderprodid\_fk REFERENCES Products,

OrderID VARCHAR2(20) CONSTRAINT postorder\_fk REFERENCES Orders,

Request\_Date DATE

);

commit;

CREATE TABLE RETURN\_CANCEL (

RequestID VARCHAR2(50) CONSTRAINT reqcanid\_fk REFERENCES POST\_ORDER\_STATUS,

Refund\_Date Date,

RefundID CONSTRAINT ret\_fk REFERENCES Refund,

Refund\_Status VARCHAR2(255)

);

commit;

CREATE TABLE EXCHANGE\_ORDERS (

RequestID VARCHAR2(50) CONSTRAINT ret\_can\_fk REFERENCES POST\_ORDER\_STATUS,

Exchange\_Resolution\_Date Date,

Exchange\_Status VARCHAR2(50) CONSTRAINT exchange\_check CHECK (Exchange\_Status IN ('Exchange', 'Pending', 'Closed', 'Cancelled')),

Exchange\_Method VARCHAR2(50) CONSTRAINT method\_check CHECK (Exchange\_Method IN ('In-store', 'Pickup'))

);

commit;

CREATE TABLE Reviews (

ReviewID VARCHAR2(50) CONSTRAINT revid\_pk PRIMARY KEY,

ProductID VARCHAR2(50) CONSTRAINT revprodid\_fk REFERENCES Products

Review VARCHAR2(255)

);

commit;

CREATE TABLE SUPPLIES (

SupplierID VARCHAR2(50) CONSTRAINT supplyshipno\_fk REFERENCES Suppliers,

ProductID VARCHAR2(50) CONSTRAINT supplyprodid\_fk REFERENCES Products

);

commit;

CREATE TABLE CUSTOMER (

CustomerID VARCHAR2(20) PRIMARY KEY,

first\_name VARCHAR2(50),

last\_name VARCHAR2(50),

contactno VARCHAR2(20),

email VARCHAR2(100),

age NUMBER,

dob DATE,

gender VARCHAR2(10));

CREATE TABLE shift\_crew (

SHIFT\_CREWID VARCHAR2(15),

START\_TIME VARCHAR2(10),

END\_TIME VARCHAR2(10),

SHIFT\_TYPE VARCHAR2(10),

PERFORMANCE\_RATINGS NUMBER(1, 0) DEFAULT 5,

CONSTRAINT RATING\_CHK CHECK (PERFORMANCE\_RATINGS BETWEEN 1 AND 5),

CONSTRAINT STYPE\_CHK CHECK (SHIFT\_TYPE IN ('Morning', 'Evening', 'Night')),

PRIMARY KEY (SHIFT\_CREWID)

);

---------------------------------------------------------------------

-- Age (Derived) attribute from DOB for Customer table

CREATE OR REPLACE TRIGGER calculate\_age\_trigger

BEFORE INSERT OR UPDATE ON CUSTOMER

FOR EACH ROW

BEGIN

:NEW.AGE := TRUNC(MONTHS\_BETWEEN(SYSDATE, :NEW.DOB) / 12);

END;

/

---------------------------------------------------------------------

-- Data insertion script

INSERT INTO Cashier (EmployeeID, Register\_Number)

SELECT Employee\_ID, ROUND(DBMS\_RANDOM.VALUE(1, 99)) AS RegisterNumber

FROM Employees

WHERE Job\_Title = 'Cashier';

INSERT INTO Attendants (EmployeeID, Security\_Access\_Level)

SELECT Employee\_ID,

(CASE

WHEN DBMS\_RANDOM.VALUE < 0.33 THEN 'Online'

WHEN DBMS\_RANDOM.VALUE BETWEEN 0.33 AND 0.66 THEN 'Physical'

ELSE 'Both' END) AS Security\_Access\_Level

FROM Employees

WHERE Job\_Title = 'Attendant';

INSERT INTO Employee\_Supervisor (SupervisorID, Employees\_Supervised)

SELECT Employee\_ID, ROUND(DBMS\_RANDOM.VALUE(1, 99)) AS Employees\_Supervised

FROM Employees

WHERE Job\_Title = 'Supervisor';

---------------------------------------------------------------------

-- Create the WAREHOUSES table

CREATE TABLE WAREHOUSES (

WarehouseCode VARCHAR(5) PRIMARY KEY,

WarehouseName VARCHAR(50),

WarehouseCapacity INT,

StreetAddress VARCHAR(50),

SuiteNumber VARCHAR(20),

City VARCHAR(50),

ZipCode VARCHAR(10),

State VARCHAR(50),

ManagerName VARCHAR(50)

);

-- Insert data into the WAREHOUSES table

INSERT INTO WAREHOUSES VALUES ('WH001', 'Central Warehouse', 50000, '123 Main St', 'Suite 101', 'Los Angeles', '90001', 'California', 'John Smith');

INSERT INTO WAREHOUSES VALUES ('WH002', 'Southern Depot', 40000, '456 Oak St', 'Unit 202', 'Houston', '77002', 'Texas', 'Jane Doe');

INSERT INTO WAREHOUSES VALUES ('WH003', 'Sunshine Warehouse', 60000, '789 Pine St', 'Apt 303', 'Miami', '33101', 'Florida', 'Mike Johnson');

INSERT INTO WAREHOUSES VALUES ('WH004', 'Big Apple Depot', 55000, '890 Cedar St', 'Bldg 5', 'New York City', '10001', 'New York', 'Emily White');

INSERT INTO WAREHOUSES VALUES ('WH005', 'Peach State Warehouse', 48000, '111 Birch St', 'Suite 15', 'Atlanta', '30301', 'Georgia', 'David Davis');

INSERT INTO WAREHOUSES VALUES ('WH006', 'Windy City Depot', 52000, '222 Elm St', 'Unit 18', 'Chicago', '60601', 'Illinois', 'Sophie Miller');

INSERT INTO WAREHOUSES VALUES ('WH007', 'Grand Canyon Warehouse', 45000, '333 Pine St', 'Bldg 3', 'Phoenix', '85001', 'Arizona', 'Andrew Brown');

INSERT INTO WAREHOUSES VALUES ('WH008', 'Mile High Depot', 48000, '444 Maple St', 'Apt 404', 'Denver', '80202', 'Colorado', 'Olivia Wilson');

INSERT INTO WAREHOUSES VALUES ('WH009', 'Queen City Depot', 55000, '555 Oak St', 'Suite 25', 'Charlotte', '28202', 'North Carolina', 'Daniel Taylor');

INSERT INTO WAREHOUSES VALUES ('WH010', 'Liberty Warehouse', 60000, '666 Birch St', 'Unit 12', 'Philadelphia', '19102', 'Pennsylvania', 'Emma Robinson');

INSERT INTO WAREHOUSES VALUES ('WH011', 'Buckeye Depot', 52000, '777 Pine St', 'Bldg 7', 'Columbus', '43201', 'Ohio', 'Jack Harris');

INSERT INTO WAREHOUSES VALUES ('WH012', 'Motor City Warehouse', 45000, '888 Cedar St', 'Apt 505', 'Detroit', '48201', 'Michigan', 'Ava Martin');

INSERT INTO WAREHOUSES VALUES ('WH013', 'Volunteer State Depot', 48000, '999 Elm St', 'Unit 15', 'Nashville', '37201', 'Tennessee', 'Noah Garcia');

INSERT INTO WAREHOUSES VALUES ('WH014', 'Emerald City Warehouse', 55000, '123 Birch St', 'Bldg 2', 'Seattle', '98101', 'Washington', 'Mia Thompson');

INSERT INTO WAREHOUSES VALUES ('WH015', 'Lone Star Depot', 60000, '234 Oak St', 'Suite 20', 'Dallas', '75201', 'Texas', 'Liam Clark');

INSERT INTO WAREHOUSES VALUES ('WH016', 'Bay State Warehouse', 52000, '345 Pine St', 'Unit 10', 'Boston', '02101', 'Massachusetts', 'Lily Turner');

INSERT INTO WAREHOUSES VALUES ('WH017', 'Show Me State Depot', 45000, '456 Maple St', 'Bldg 8', 'Kansas City', '64101', 'Missouri', 'Ethan Baker');

INSERT INTO WAREHOUSES VALUES ('WH018', 'Garden State Warehouse', 48000, '567 Elm St', 'Apt 303', 'Jersey City', '07302', 'New Jersey', 'Grace Hall');

INSERT INTO WAREHOUSES VALUES ('WH019', 'Land of Lakes Depot', 55000, '678 Cedar St', 'Bldg 6', 'Minneapolis', '55401', 'Minnesota', 'Logan Harris');

INSERT INTO WAREHOUSES VALUES ('WH020', 'Old Dominion Warehouse', 60000, '789 Birch St', 'Suite 18', 'Richmond', '23218', 'Virginia', 'Aiden Lee');

INSERT INTO WAREHOUSES VALUES ('WH021', 'Sunshine State Depot', 52000, '890 Oak St', 'Unit 303', 'Orlando', '32801', 'Florida', 'Sofia Young');

INSERT INTO WAREHOUSES VALUES ('WH022', 'Alamo Warehouse', 45000, '111 Pine St', 'Bldg 4', 'San Antonio', '78201', 'Texas', 'Carter Smith');

INSERT INTO WAREHOUSES VALUES ('WH023', 'Golden Gate Depot', 48000, '222 Birch St', 'Apt 404', 'San Francisco', '94102', 'California', 'Madison Davis');

INSERT INTO WAREHOUSES VALUES ('WH024', 'Prairie State Warehouse', 55000, '333 Cedar St', 'Suite 22', 'Springfield', '62701', 'Illinois', 'Lucas Miller');

INSERT INTO WAREHOUSES VALUES ('WH025', 'Mile High City Depot', 60000, '444 Elm St', 'Unit 12', 'Colorado Springs', '80901', 'Colorado', 'Zoe Turner');

INSERT INTO WAREHOUSES VALUES ('WH026', 'Empire State Warehouse', 52000, '555 Oak St', 'Bldg 5', 'New York', '10001', 'New York', 'Jackson Harris');

INSERT INTO WAREHOUSES VALUES ('WH027', 'Peach State Depot', 45000, '666 Pine St', 'Apt 303', 'Atlanta', '30301', 'Georgia', 'Chloe White');

INSERT INTO WAREHOUSES VALUES ('WH028', 'Grand Canyon Warehouse', 48000, '777 Birch St', 'Suite 25', 'Phoenix', '85001', 'Arizona', 'Henry Johnson');

INSERT INTO WAREHOUSES VALUES ('WH029', 'Old Dominion Depot', 55000, '888 Maple St', 'Bldg 3', 'Richmond', '23218', 'Virginia', 'Ella Wilson');

INSERT INTO WAREHOUSES VALUES ('WH030', 'Tar Heel State Warehouse', 60000, '999 Elm St', 'Unit 15', 'Charlotte', '28202', 'North Carolina', 'Gabriel Robinson');