

SportSphere:

An Online Sports Equipment Store

Tung Duc Vu (2023558), Gaurav Shankar (2023219), Aditya Chaudhary (2023042), Turbold Amarbat (2023559)



Deadline 1: Project Scope and Requirements

SportSphere is an online sports equipment store with dedicated portals for customers, admins, and delivery agents. It offers secure login, real-time tracking, inventory management, and performance analytics. Built using MySQL and Python tkinter for the UI, the system ensures scalability, data integrity, and a seamless shopping experience with efficient delivery and feedback systems.

01

Project Scope

SportSphere is an online sports equipment store designed to streamline shopping for customers and simplify operations for admins and delivery agents. It supports secure logins, product browsing, order management, real-time delivery tracking, and customer feedback. Admins manage inventory and analyze performance, while agents handle timely deliveries. Built on a structured ER-based DBMS, the system emphasizes scalability, data integrity, and accessibility to ensure a reliable and efficient platform for all stakeholders.

02

Technical requirements

SportSphere is built using a modern tech stack to ensure high performance, scalability, and security:

- Database Management: MySQL
- Backend Development: Python
- Frontend Development: HTML, CSS, JavaScript, ReactJS

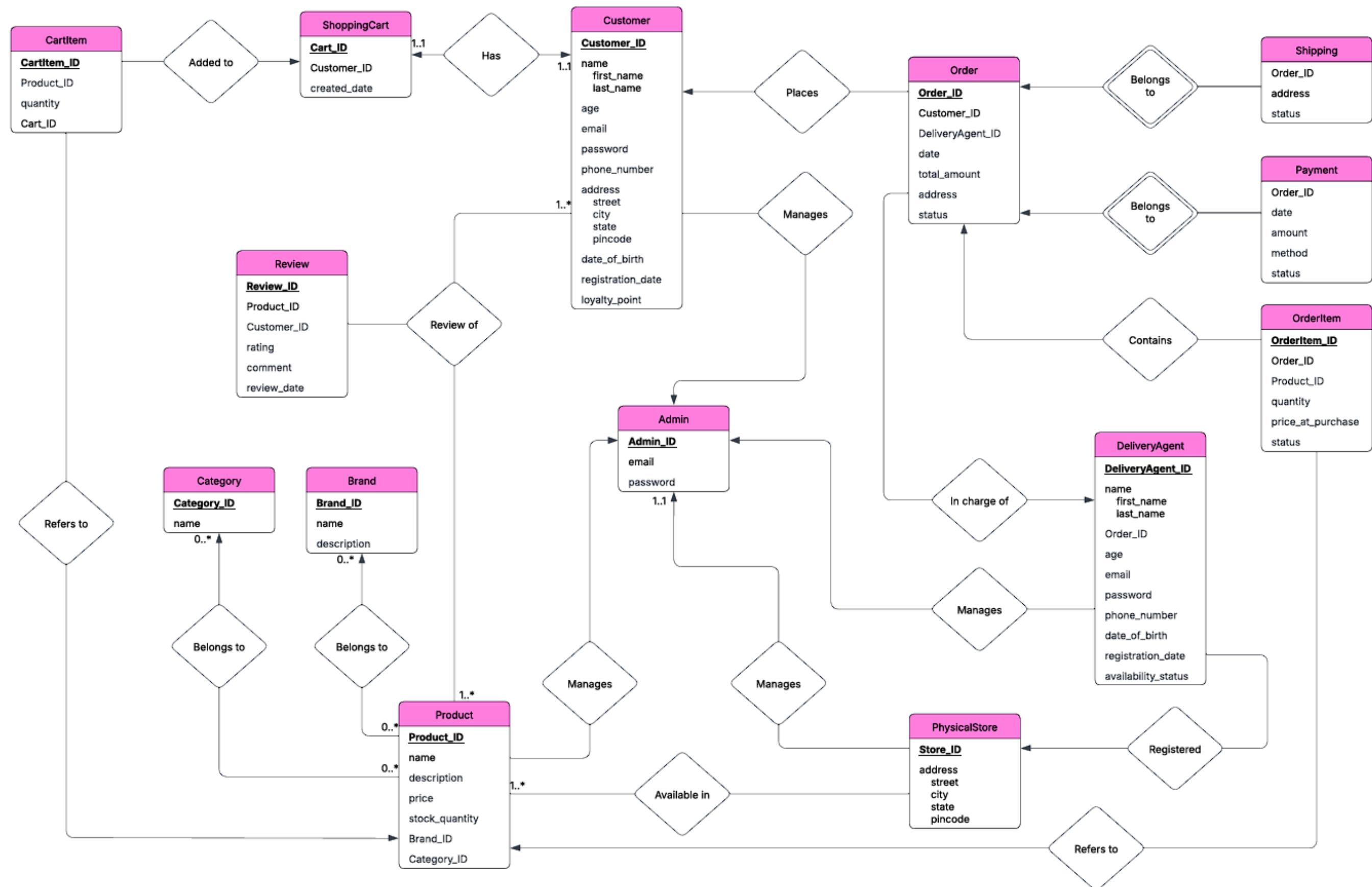
03

System Functionality

SportSphere supports secure signup and login for customers, admins, and delivery agents. Customers can browse products, use a shopping cart, track orders in real-time, earn loyalty points, and submit reviews. Admins manage inventory, store operations, and delivery agents, while analyzing orders, sales, and customer feedback. Delivery partners receive order assignments based on availability, provide live status updates, and track tasks via a dashboard. The system ensures smooth coordination across all roles for efficient store management and a seamless user experience.



Deadline 1: ER Diagram



The ER model outlines key entities in SportSphere: Customers, Admins, Delivery Agents, and Physical Stores. Customers can browse products (linked to categories and brands), manage carts, place orders, and leave reviews. Orders contain items, are linked to payments and shipping, and are managed by delivery agents. Admins oversee inventory, store operations, and agent performance. Delivery agents update real-time order status. The design ensures efficient data flow, scalability, and seamless coordination among system components to support all core functionalities of the platform.



Deadline 2: Relational Model

01

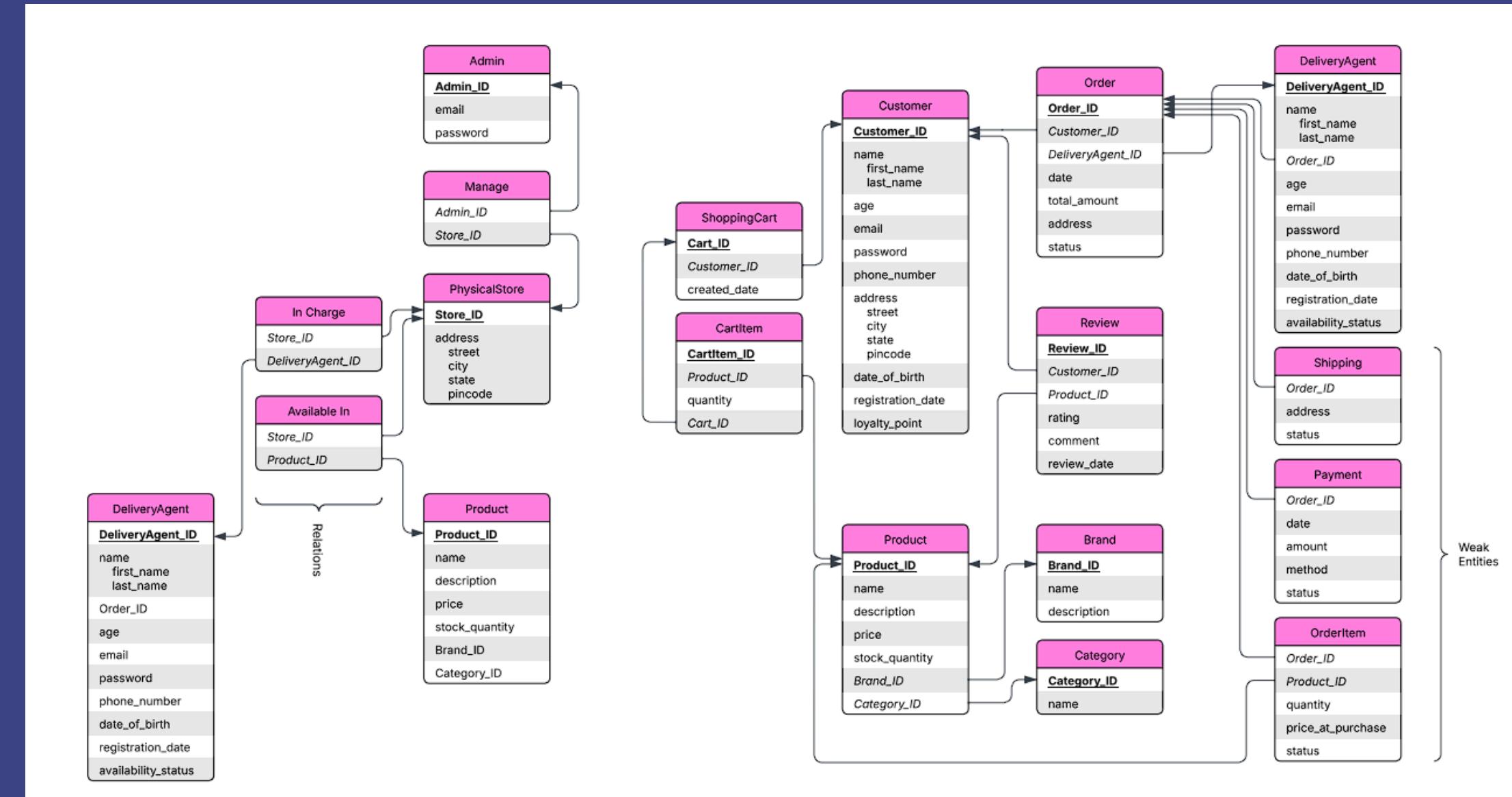
Customer-Centric Experience:

Customers can securely register, shop using a cart system, place and track orders, and give product reviews. All relationships are well-connected, ensuring a seamless shopping experience and reliable order lifecycle management from cart to delivery.

02

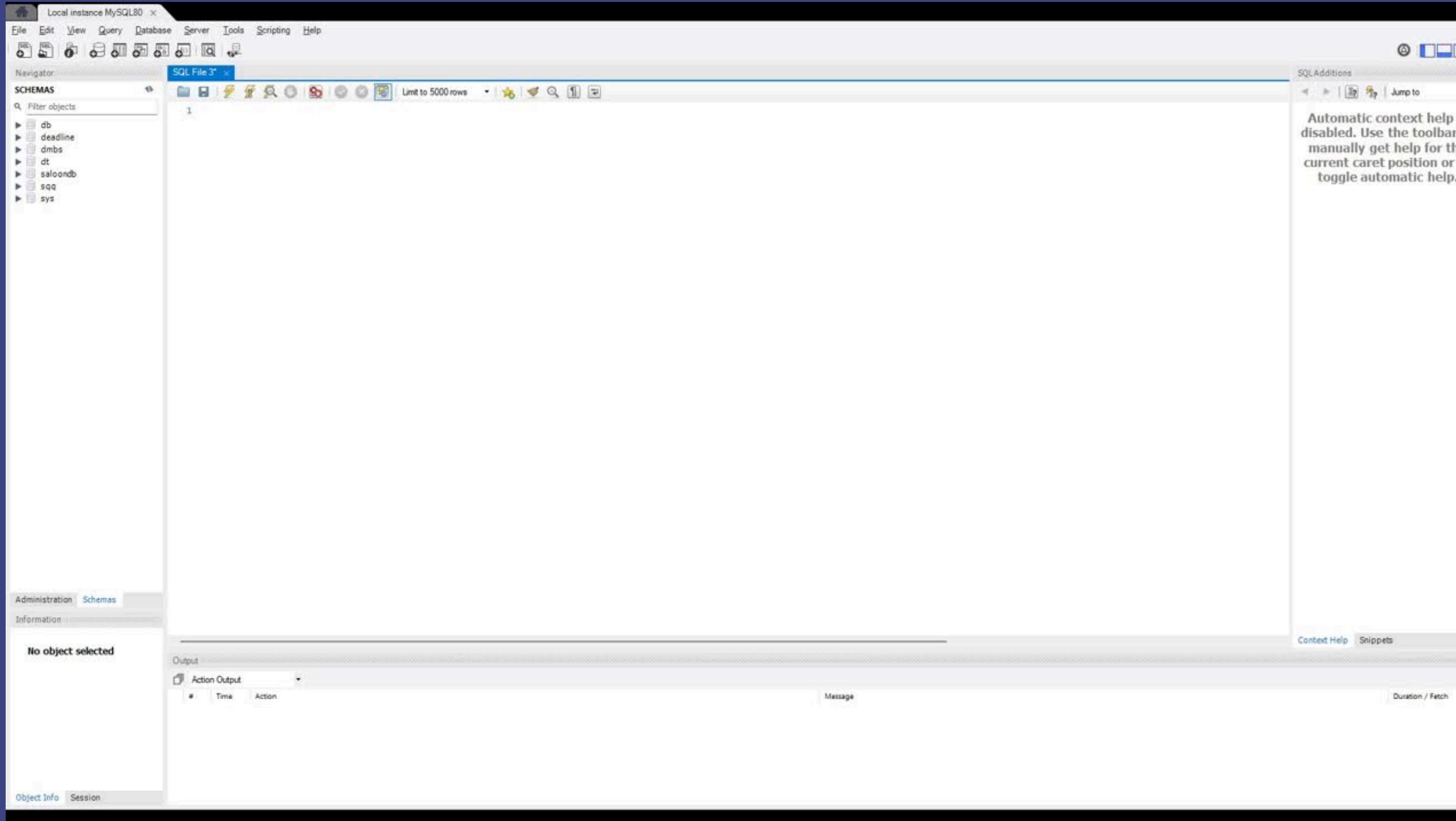
Enhanced Operational Flow:

Admins manage physical stores, while Delivery Agents are assigned to stores via the In Charge relationship. The Available In relation between stores and products allows real-time inventory tracking across locations, supporting smooth order fulfillment.





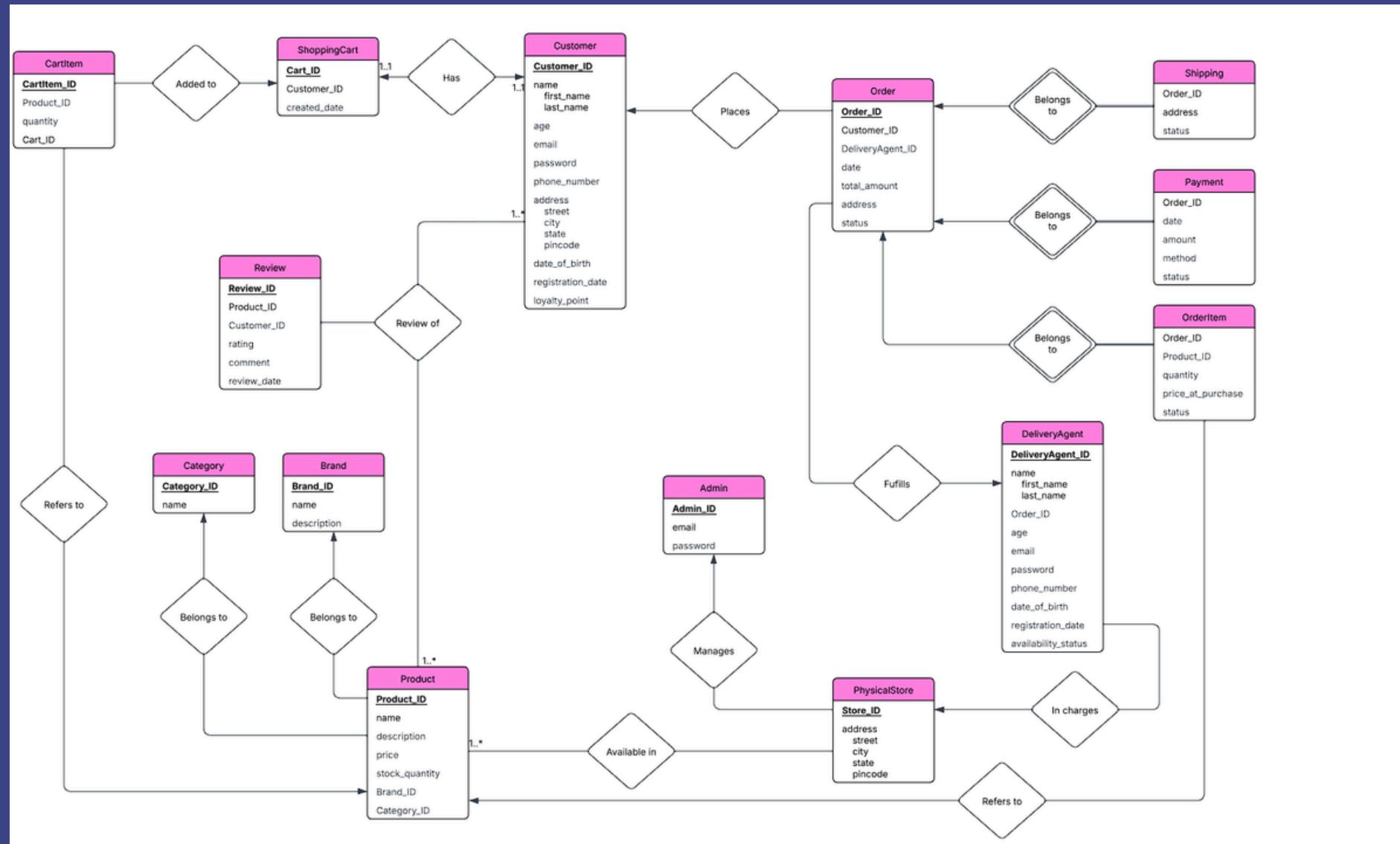
Deadline2: SQL



We designed and created the `sports_store_db` database to manage an online and offline sports equipment store. It includes core entities like Customer, Admin, Product, Brand, Category, DeliveryAgent, and PhysicalStore, with relationships modeled using foreign keys and associative tables such as OrderItem, Available_in, Manages, and InCharges. Constraints like CHECK, UNIQUE, and NOT NULL ensure data integrity. We populated the tables by executing SQL scripts in MySQL Workbench, using INSERT INTO statements with sample data for customers, products, and store info to test all functionalities.



Refinement: ER Model



01

Customer Flow: Customers manage carts, place orders, track deliveries, write reviews, and earn loyalty points. Each order links to payment, shipping, and delivery status.

02

Admin Oversight: Admins manage inventory, stores, and monitor product availability, ensuring operational efficiency.

03

Delivery Fulfillment: Delivery agents are assigned to orders and update fulfillment status. Products belong to categories and brands, and are stocked in physical stores.

Main Changes from the First ER Model:

The updated ER diagram introduces a more streamlined structure with clearer relationships and fewer redundancies. The “Manages” relationships are now simplified — Admins manage stores directly, and Delivery Agents fulfill orders instead of being in charge of multiple entities. The “Order” entity is now more centralized, linking directly to OrderItem, Payment, and Shipping, ensuring modular tracking of each order component. Additionally, “PhysicalStore” is better integrated to represent stock availability, and the “Product” entity is now more focused with direct relations to Brand and Category. These refinements improve clarity, data integrity, and system efficiency.



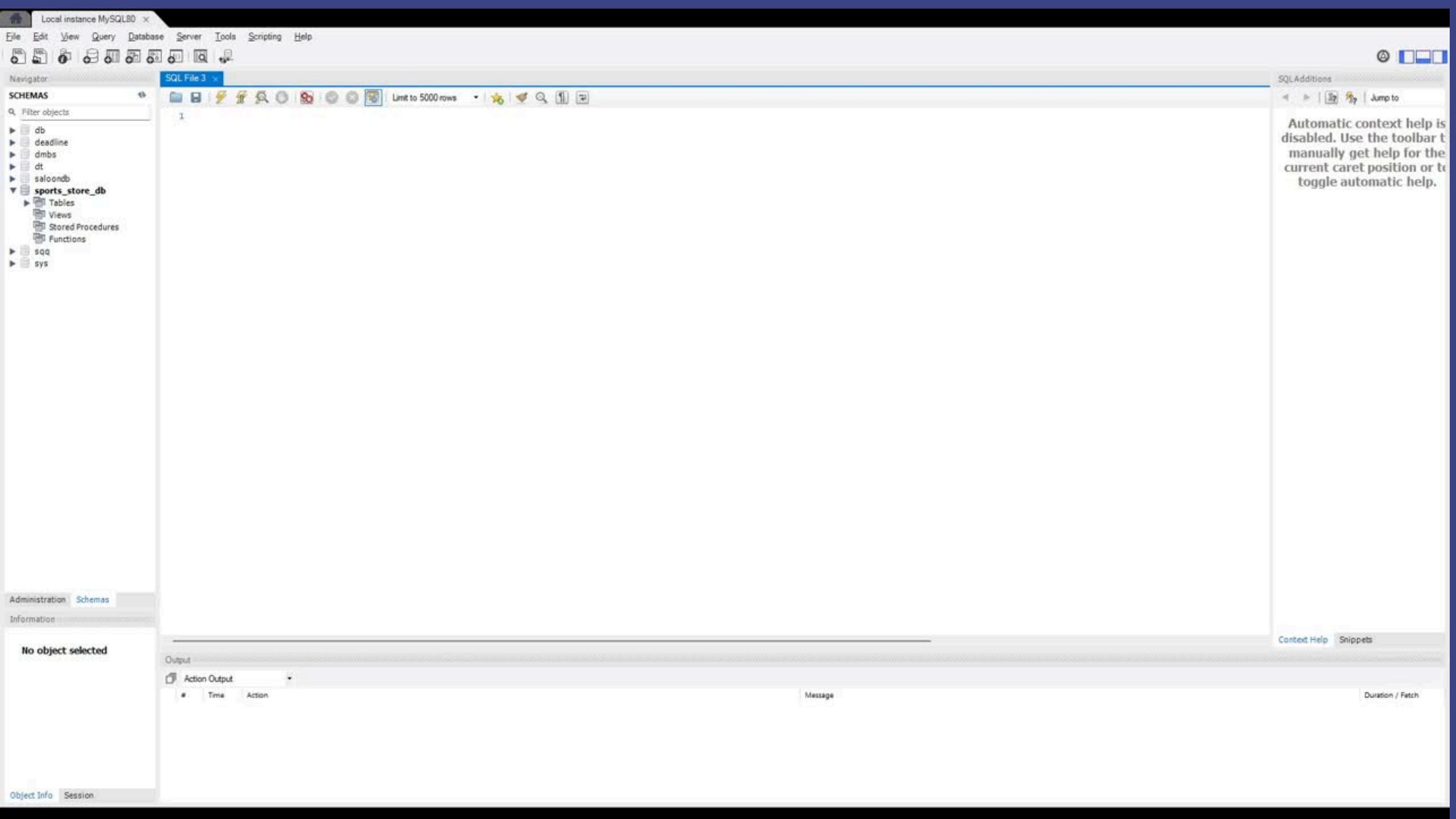
Deadline 3: Functional Queries

```

query.sql + 43cnmpkmg
STON Input for the program (Optional)
541 -- 1. Top 10 Customers by Total_Spent
542 SELECT c.Customer_ID, c.first_name, c.last_name, SUM(o.total_price) AS Total_Spent
543 FROM Customer c
544 JOIN orders o ON c.Customer_ID = o.Customer_ID
545 JOIN order_items oi ON o.Order_ID = oi.Order_ID
546 ORDER BY Total_Spent DESC
547 LIMIT 10;
548
549 -- 2. Top 10 Products by Revenue Generated
550 SELECT p.Product_ID, p.name, SUM(oi.quantity * oi.price_at_purchase) AS Total_Revenue
551 FROM Product p
552 JOIN order_items oi ON p.Product_ID = oi.Product_ID
553 JOIN orders o ON oi.Order_ID = o.Order_ID
554 ORDER BY Total_Revenue DESC
555 LIMIT 10;
556
557 -- 3. Top 10 Most Ordered Products
558 SELECT p.Product_ID, p.name, SUM(oi.quantity) AS Total_Ordered
559 FROM Order_Items oi
560 JOIN Product p ON oi.Product_ID = p.Product_ID
561 GROUP BY p.Product_ID, p.name
562 ORDER BY Total_Ordered DESC
563 LIMIT 10;
564
565 -- 4. Top 10 Delivery Agents by Rating
566 SELECT da.DeliveryAgent_ID, da.first_name, da.last_name, AVG(dr.rating) AS Average_Rating
567 FROM Delivery_Review dr
568 JOIN Delivery_Agent da ON dr.DeliveryAgent_ID = da.DeliveryAgent_ID
569 JOIN orders o ON da.DeliveryAgent_ID = o.DeliveryAgent_ID
570 JOIN order_items oi ON o.Order_ID = oi.Order_ID
571 ORDER BY Average_Rating DESC
572 LIMIT 10;
573
574 -- 5. Average Order Costs, Total Number of Orders, Revenue per Month
575 DATE_FORMAT(date, 'YY-Mm') AS Month,
576 COUNT(o.Order_ID) AS Total_Orders,
577 SUM(o.total_price) AS Total_Revenue,
578 AVG(o.total_price) AS Avg_Order_Cost
579 WHERE o.status = 'Delivered'
580
581 -- 6. Top 10 Most Reviewed Products
582 SELECT p.Product_ID, p.name, COUNT(r.Product_ID) AS Total_Reviews
583 FROM Product p
584 JOIN Product_Review r ON p.Product_ID = r.Product_ID
585 GROUP BY p.Product_ID, p.name
586 ORDER BY Total_Reviews DESC
587 LIMIT 10;
588
589 -- 7. Update Product Details
590 UPDATE Product
591 SET description = 'Updated description for the product.'
592 WHERE Product_ID = 1;
593
594 -- 8. Low Stock Alert
595 SELECT s.Store_ID, p.Product_ID, p.name, s.store_quantity
596 FROM Store s
597 WHERE s.store_quantity <= 5;
598
599 -- 9. View Order History
600 SELECT o.Order_ID, o.date, o.total_price, o.address, o.status
601 FROM orders o
602 WHERE o.Customer_ID = 1;
603
604 -- 10. Create Trigger for Price Drop Alerts
605 CREATE TRIGGER price_drop_alert
606 AFTER UPDATE ON Product
607 FOR EACH ROW
608 BEGIN
609   IF NEW.price < OLD.price THEN
610     INSERT INTO Product_History (Customer_ID, Product_ID, old_price, new_price, changed_date)
611     VALUES (1, NEW.Product_ID, OLD.price, NEW.price, NOW());
612 END IF;
613 END;
614
615 -- 11. Get Feedback on Deliveries
616 SELECT AVG(dr.rating) AS Average_Rating
617 FROM Delivery_Review dr
618 WHERE dr.DeliveryAgent_ID = 1;
619
620 -- 12. Get Feedback on Orders
621 SELECT SUM(o.total_price) AS total_revenue
622 FROM orders o
623 WHERE DeliveryAgent_ID = 1 AND YEAR(date) = 2024;
624
625 -- 13. Get Feedback on Delivery Agents
626 SELECT SUM(dr.rating) AS average_rating
627 FROM Delivery_Review dr
628 WHERE dr.DeliveryAgent_ID = 1;
629
630 -- 14. Get Feedback on Products
631 SELECT SUM(pr.rating) AS average_review
632 FROM Product_Review pr
633 WHERE pr.Product_ID = 1;
634
635 -- 15. Get Feedback on Categories
636 SELECT SUM(c.rating) AS average_rating
637 FROM Category_Collection cc
638 JOIN Category c ON cc.Category_ID = c.Category_ID
639 WHERE cc.Customer_ID = 1;
640
641 -- 16. Get Feedback on Products
642 SELECT SUM(pr.rating) AS average_review
643 FROM Product_Review pr
644 WHERE pr.Product_ID = 1;
645
646 -- 17. Get Feedback on Delivery Agents
647 SELECT SUM(dr.rating) AS average_rating
648 FROM Delivery_Review dr
649 WHERE dr.DeliveryAgent_ID = 1;
650
651 -- 18. Get Feedback on Products
652 SELECT SUM(pr.rating) AS average_review
653 FROM Product_Review pr
654 WHERE pr.Product_ID = 1;
655
656 -- 19. Get Feedback on Delivery Agents
657 SELECT SUM(dr.rating) AS average_rating
658 FROM Delivery_Review dr
659 WHERE dr.DeliveryAgent_ID = 1;
660
661 -- 20. Get Feedback on Products
662 SELECT SUM(pr.rating) AS average_review
663 FROM Product_Review pr
664 WHERE pr.Product_ID = 1;
665
666 -- 21. Get Feedback on Delivery Agents
667 SELECT SUM(dr.rating) AS average_rating
668 FROM Delivery_Review dr
669 WHERE dr.DeliveryAgent_ID = 1;
670
671 -- 22. Get Feedback on Products
672 SELECT SUM(pr.rating) AS average_review
673 FROM Product_Review pr
674 WHERE pr.Product_ID = 1;
675
676 -- 23. Get Feedback on Delivery Agents
677 SELECT SUM(dr.rating) AS average_rating
678 FROM Delivery_Review dr
679 WHERE dr.DeliveryAgent_ID = 1;
680
681 -- 24. Get Feedback on Products
682 SELECT SUM(pr.rating) AS average_review
683 FROM Product_Review pr
684 WHERE pr.Product_ID = 1;
685
686 -- 25. Get Feedback on Delivery Agents
687 SELECT SUM(dr.rating) AS average_rating
688 FROM Delivery_Review dr
689 WHERE dr.DeliveryAgent_ID = 1;
690
691 -- 26. Get Feedback on Products
692 SELECT SUM(pr.rating) AS average_review
693 FROM Product_Review pr
694 WHERE pr.Product_ID = 1;
695
696 -- 27. Get Feedback on Delivery Agents
697 SELECT SUM(dr.rating) AS average_rating
698 FROM Delivery_Review dr
699 WHERE dr.DeliveryAgent_ID = 1;
700
701 -- 28. Get Feedback on Products
702 SELECT SUM(pr.rating) AS average_review
703 FROM Product_Review pr
704 WHERE pr.Product_ID = 1;
705
706 -- 29. Get Feedback on Delivery Agents
707 SELECT SUM(dr.rating) AS average_rating
708 FROM Delivery_Review dr
709 WHERE dr.DeliveryAgent_ID = 1;
710
711 -- 30. Get Feedback on Products
712 SELECT SUM(pr.rating) AS average_review
713 FROM Product_Review pr
714 WHERE pr.Product_ID = 1;
715
716 -- 31. Get Feedback on Delivery Agents
717 SELECT SUM(dr.rating) AS average_rating
718 FROM Delivery_Review dr
719 WHERE dr.DeliveryAgent_ID = 1;
720
721 -- 32. Get Feedback on Products
722 SELECT SUM(pr.rating) AS average_review
723 FROM Product_Review pr
724 WHERE pr.Product_ID = 1;
725
726 -- 33. Get Feedback on Delivery Agents
727 SELECT SUM(dr.rating) AS average_rating
728 FROM Delivery_Review dr
729 WHERE dr.DeliveryAgent_ID = 1;
730
731 -- 34. Get Feedback on Products
732 SELECT SUM(pr.rating) AS average_review
733 FROM Product_Review pr
734 WHERE pr.Product_ID = 1;
735
736 -- 35. Get Feedback on Delivery Agents
737 SELECT SUM(dr.rating) AS average_rating
738 FROM Delivery_Review dr
739 WHERE dr.DeliveryAgent_ID = 1;
740
741 -- 36. Get Feedback on Products
742 SELECT SUM(pr.rating) AS average_review
743 FROM Product_Review pr
744 WHERE pr.Product_ID = 1;
745
746 -- 37. Get Feedback on Delivery Agents
747 SELECT SUM(dr.rating) AS average_rating
748 FROM Delivery_Review dr
749 WHERE dr.DeliveryAgent_ID = 1;
750
751 -- 38. Get Feedback on Products
752 SELECT SUM(pr.rating) AS average_review
753 FROM Product_Review pr
754 WHERE pr.Product_ID = 1;
755
756 -- 39. Get Feedback on Delivery Agents
757 SELECT SUM(dr.rating) AS average_rating
758 FROM Delivery_Review dr
759 WHERE dr.DeliveryAgent_ID = 1;
760
761 -- 40. Get Feedback on Products
762 SELECT SUM(pr.rating) AS average_review
763 FROM Product_Review pr
764 WHERE pr.Product_ID = 1;
765
766 -- 41. Get Feedback on Delivery Agents
767 SELECT SUM(dr.rating) AS average_rating
768 FROM Delivery_Review dr
769 WHERE dr.DeliveryAgent_ID = 1;
770
771 -- 42. Get Feedback on Products
772 SELECT SUM(pr.rating) AS average_review
773 FROM Product_Review pr
774 WHERE pr.Product_ID = 1;
775
776 -- 43. Get Feedback on Delivery Agents
777 SELECT SUM(dr.rating) AS average_rating
778 FROM Delivery_Review dr
779 WHERE dr.DeliveryAgent_ID = 1;
780
781 -- 44. Get Feedback on Products
782 SELECT SUM(pr.rating) AS average_review
783 FROM Product_Review pr
784 WHERE pr.Product_ID = 1;
785
786 -- 45. Get Feedback on Delivery Agents
787 SELECT SUM(dr.rating) AS average_rating
788 FROM Delivery_Review dr
789 WHERE dr.DeliveryAgent_ID = 1;
790
791 -- 46. Get Feedback on Products
792 SELECT SUM(pr.rating) AS average_review
793 FROM Product_Review pr
794 WHERE pr.Product_ID = 1;
795
796 -- 47. Get Feedback on Delivery Agents
797 SELECT SUM(dr.rating) AS average_rating
798 FROM Delivery_Review dr
799 WHERE dr.DeliveryAgent_ID = 1;
800
801 -- 48. Get Feedback on Products
802 SELECT SUM(pr.rating) AS average_review
803 FROM Product_Review pr
804 WHERE pr.Product_ID = 1;
805
806 -- 49. Get Feedback on Delivery Agents
807 SELECT SUM(dr.rating) AS average_rating
808 FROM Delivery_Review dr
809 WHERE dr.DeliveryAgent_ID = 1;
810
811 -- 50. Get Feedback on Products
812 SELECT SUM(pr.rating) AS average_review
813 FROM Product_Review pr
814 WHERE pr.Product_ID = 1;
815
816 -- 51. Get Feedback on Delivery Agents
817 SELECT SUM(dr.rating) AS average_rating
818 FROM Delivery_Review dr
819 WHERE dr.DeliveryAgent_ID = 1;
820
821 -- 52. Get Feedback on Products
822 SELECT SUM(pr.rating) AS average_review
823 FROM Product_Review pr
824 WHERE pr.Product_ID = 1;
825
826 -- 53. Get Feedback on Delivery Agents
827 SELECT SUM(dr.rating) AS average_rating
828 FROM Delivery_Review dr
829 WHERE dr.DeliveryAgent_ID = 1;
830
831 -- 54. Get Feedback on Products
832 SELECT SUM(pr.rating) AS average_review
833 FROM Product_Review pr
834 WHERE pr.Product_ID = 1;
835
836 -- 55. Get Feedback on Delivery Agents
837 SELECT SUM(dr.rating) AS average_rating
838 FROM Delivery_Review dr
839 WHERE dr.DeliveryAgent_ID = 1;
840
841 -- 56. Get Feedback on Products
842 SELECT SUM(pr.rating) AS average_review
843 FROM Product_Review pr
844 WHERE pr.Product_ID = 1;
845
846 -- 57. Get Feedback on Delivery Agents
847 SELECT SUM(dr.rating) AS average_rating
848 FROM Delivery_Review dr
849 WHERE dr.DeliveryAgent_ID = 1;
850
851 -- 58. Get Feedback on Products
852 SELECT SUM(pr.rating) AS average_review
853 FROM Product_Review pr
854 WHERE pr.Product_ID = 1;
855
856 -- 59. Get Feedback on Delivery Agents
857 SELECT SUM(dr.rating) AS average_rating
858 FROM Delivery_Review dr
859 WHERE dr.DeliveryAgent_ID = 1;
860
861 -- 60. Get Feedback on Products
862 SELECT SUM(pr.rating) AS average_review
863 FROM Product_Review pr
864 WHERE pr.Product_ID = 1;
865
866 -- 61. Get Feedback on Delivery Agents
867 SELECT SUM(dr.rating) AS average_rating
868 FROM Delivery_Review dr
869 WHERE dr.DeliveryAgent_ID = 1;
870
871 -- 62. Get Feedback on Products
872 SELECT SUM(pr.rating) AS average_review
873 FROM Product_Review pr
874 WHERE pr.Product_ID = 1;
875
876 -- 63. Get Feedback on Delivery Agents
877 SELECT SUM(dr.rating) AS average_rating
878 FROM Delivery_Review dr
879 WHERE dr.DeliveryAgent_ID = 1;
880
881 -- 64. Get Feedback on Products
882 SELECT SUM(pr.rating) AS average_review
883 FROM Product_Review pr
884 WHERE pr.Product_ID = 1;
885
886 -- 65. Get Feedback on Delivery Agents
887 SELECT SUM(dr.rating) AS average_rating
888 FROM Delivery_Review dr
889 WHERE dr.DeliveryAgent_ID = 1;
890
891 -- 66. Get Feedback on Products
892 SELECT SUM(pr.rating) AS average_review
893 FROM Product_Review pr
894 WHERE pr.Product_ID = 1;
895
896 -- 67. Get Feedback on Delivery Agents
897 SELECT SUM(dr.rating) AS average_rating
898 FROM Delivery_Review dr
899 WHERE dr.DeliveryAgent_ID = 1;
900
901 -- 68. Get Feedback on Products
902 SELECT SUM(pr.rating) AS average_review
903 FROM Product_Review pr
904 WHERE pr.Product_ID = 1;
905
906 -- 69. Get Feedback on Delivery Agents
907 SELECT SUM(dr.rating) AS average_rating
908 FROM Delivery_Review dr
909 WHERE dr.DeliveryAgent_ID = 1;
910
911 -- 70. Get Feedback on Products
912 SELECT SUM(pr.rating) AS average_review
913 FROM Product_Review pr
914 WHERE pr.Product_ID = 1;
915
916 -- 71. Get Feedback on Delivery Agents
917 SELECT SUM(dr.rating) AS average_rating
918 FROM Delivery_Review dr
919 WHERE dr.DeliveryAgent_ID = 1;
920
921 -- 72. Get Feedback on Products
922 SELECT SUM(pr.rating) AS average_review
923 FROM Product_Review pr
924 WHERE pr.Product_ID = 1;
925
926 -- 73. Get Feedback on Delivery Agents
927 SELECT SUM(dr.rating) AS average_rating
928 FROM Delivery_Review dr
929 WHERE dr.DeliveryAgent_ID = 1;
930
931 -- 74. Get Feedback on Products
932 SELECT SUM(pr.rating) AS average_review
933 FROM Product_Review pr
934 WHERE pr.Product_ID = 1;
935
936 -- 75. Get Feedback on Delivery Agents
937 SELECT SUM(dr.rating) AS average_rating
938 FROM Delivery_Review dr
939 WHERE dr.DeliveryAgent_ID = 1;
940
941 -- 76. Get Feedback on Products
942 SELECT SUM(pr.rating) AS average_review
943 FROM Product_Review pr
944 WHERE pr.Product_ID = 1;
945
946 -- 77. Get Feedback on Delivery Agents
947 SELECT SUM(dr.rating) AS average_rating
948 FROM Delivery_Review dr
949 WHERE dr.DeliveryAgent_ID = 1;
950
951 -- 78. Get Feedback on Products
952 SELECT SUM(pr.rating) AS average_review
953 FROM Product_Review pr
954 WHERE pr.Product_ID = 1;
955
956 -- 79. Get Feedback on Delivery Agents
957 SELECT SUM(dr.rating) AS average_rating
958 FROM Delivery_Review dr
959 WHERE dr.DeliveryAgent_ID = 1;
960
961 -- 80. Get Feedback on Products
962 SELECT SUM(pr.rating) AS average_review
963 FROM Product_Review pr
964 WHERE pr.Product_ID = 1;
965
966 -- 81. Get Feedback on Delivery Agents
967 SELECT SUM(dr.rating) AS average_rating
968 FROM Delivery_Review dr
969 WHERE dr.DeliveryAgent_ID = 1;
970
971 -- 82. Get Feedback on Products
972 SELECT SUM(pr.rating) AS average_review
973 FROM Product_Review pr
974 WHERE pr.Product_ID = 1;
975
976 -- 83. Get Feedback on Delivery Agents
977 SELECT SUM(dr.rating) AS average_rating
978 FROM Delivery_Review dr
979 WHERE dr.DeliveryAgent_ID = 1;
980
981 -- 84. Get Feedback on Products
982 SELECT SUM(pr.rating) AS average_review
983 FROM Product_Review pr
984 WHERE pr.Product_ID = 1;
985
986 -- 85. Get Feedback on Delivery Agents
987 SELECT SUM(dr.rating) AS average_rating
988 FROM Delivery_Review dr
989 WHERE dr.DeliveryAgent_ID = 1;
990
991 -- 86. Get Feedback on Products
992 SELECT SUM(pr.rating) AS average_review
993 FROM Product_Review pr
994 WHERE pr.Product_ID = 1;
995
996 -- 87. Get Total Cart Value
997 SELECT SUM(p.price * ac.quantity) AS Total_Cart_Value
998 FROM Add_to_cart ac
999 JOIN Product p ON ac.Product_ID = p.Product_ID
1000 WHERE ac.Customer_ID = 1;
1001
1002 -- 8. Order: Place an Order
1003 INSERT INTO orders (Customer_ID, date, total_price, address, status)
1004 VALUES (1, '2024-01-01', 100.00, 'Mumbai, India', 'Pending');
1005
1006 -- 9. Search Product by Brand
1007 SELECT p.Product_ID, p.name, p.price, b.Brand_Name, c.Category_Name
1008 FROM Product p
1009 JOIN Brand b ON p.Brand_ID = b.Brand_ID
1010 JOIN Category c ON p.Category_ID = c.Category_ID
1011 WHERE b.Brand_Name = 'Adidas';
1012
1013 -- 10. Track Order Status
1014 SELECT o.Order_ID, o.date, o.total_price, o.address, o.status
1015 FROM orders o
1016 WHERE o.Customer_ID = 1;
1017
1018 -- 11. View Order History
1019 SELECT * FROM orders
1020 WHERE Customer_ID = 1;
1021
1022 -- 12. Update Product Details
1023 UPDATE Product
1024 SET description = 'Updated description for the product.'
1025 WHERE Product_ID = 1;
1026
1027 -- 13. Get Feedback on Products
1028 SELECT SUM(pr.rating) AS average_review
1029 FROM Product_Review pr
1030 WHERE pr.Product_ID = 1;
1031
1032 -- 14. Get Feedback on Delivery Agents
1033 SELECT SUM(dr.rating) AS average_rating
1034 FROM Delivery_Review dr
1035 WHERE dr.DeliveryAgent_ID = 1;
1036
1037 -- 15. Get Feedback on Categories
1038 SELECT SUM(c.rating) AS average_rating
1039 FROM Category_Collection cc
1040 JOIN Category c ON cc.Category_ID = c.Category_ID
1041 WHERE cc.Customer_ID = 1;
1042
1043 -- 16. Get Feedback on Products
1044 SELECT SUM(pr.rating) AS average_review
1045 FROM Product_Review pr
1046 WHERE pr.Product_ID = 1;
1047
1048 -- 17. Get Feedback on Delivery Agents
1049 SELECT SUM(dr.rating) AS average_rating
1050 FROM Delivery_Review dr
1051 WHERE dr.DeliveryAgent_ID = 1;
1052
1053 -- 18. Get Feedback on Products
1054 SELECT SUM(pr.rating) AS average_review
1055 FROM Product_Review pr
1056 WHERE pr.Product_ID = 1;
1057
1058 -- 19. Get Feedback on Delivery Agents
1059 SELECT SUM(dr.rating) AS average_rating
1060 FROM Delivery_Review dr
1061 WHERE dr.DeliveryAgent_ID = 1;
1062
1063 -- 20. Get Feedback on Products
1064 SELECT SUM(pr.rating) AS average_review
1065 FROM Product_Review pr
1066 WHERE pr.Product_ID = 1;
1067
1068 -- 21. Get Feedback on Delivery Agents
1069 SELECT SUM(dr.rating) AS average_rating
1070 FROM Delivery_Review dr
1071 WHERE dr.DeliveryAgent_ID = 1;
1072
1073 -- 22. Get Feedback on Products
1074 SELECT SUM(pr.rating) AS average_review
1075 FROM Product_Review pr
1076 WHERE pr.Product_ID = 1;
1077
1078 -- 23. Get Feedback on Delivery Agents
1079 SELECT SUM(dr.rating) AS average_rating
1080 FROM Delivery_Review dr
1081 WHERE dr.DeliveryAgent_ID = 1;
1082
1083 -- 24. Get Feedback on Products
1084 SELECT SUM(pr.rating) AS average_review
1085 FROM Product_Review pr
1086 WHERE pr.Product_ID = 1;
1087
1088 -- 25. Get Feedback on Delivery Agents
1089 SELECT SUM(dr.rating) AS average_rating
1090 FROM Delivery_Review dr
1091 WHERE dr.DeliveryAgent_ID = 1;
1092
1093 -- 26. Get Feedback on Products
1094 SELECT SUM(pr.rating) AS average_review
1095 FROM Product_Review pr
1096 WHERE pr.Product_ID = 1;
1097
1098 -- 27. Get Feedback on Delivery Agents
1099 SELECT SUM(dr.rating) AS average_rating
1100 FROM Delivery_Review dr
1101 WHERE dr.DeliveryAgent_ID = 1;
1102
1103 -- 28. Get Feedback on Products
1104 SELECT SUM(pr.rating) AS average_review
1105 FROM Product_Review pr
1106 WHERE pr.Product_ID = 1;
1107
1108 -- 29. Get Feedback on Delivery Agents
1109 SELECT SUM(dr.rating) AS average_rating
1110 FROM Delivery_Review dr
1111 WHERE dr.DeliveryAgent_ID = 1;
1112
1113 -- 30. Get Feedback on Products
1114 SELECT SUM(pr.rating) AS average_review
1115 FROM Product_Review pr
1116 WHERE pr.Product_ID = 1;
1117
1118 -- 31. Get Feedback on Delivery Agents
1119 SELECT SUM(dr.rating) AS average_rating
1120 FROM Delivery_Review dr
1121 WHERE dr.DeliveryAgent_ID = 1;
1122
1123 -- 32. Get Feedback on Products
1124 SELECT SUM(pr.rating) AS average_review
1125 FROM Product_Review pr
1126 WHERE pr.Product_ID = 1;
1127
1128 -- 33. Get Feedback on Delivery Agents
1129 SELECT SUM(dr.rating) AS average_rating
1130 FROM Delivery_Review dr
1131 WHERE dr.DeliveryAgent_ID = 1;
1132
1133 -- 34. Get Feedback on Products
1134 SELECT SUM(pr.rating) AS average_review
1135 FROM Product_Review pr
1136 WHERE pr.Product_ID = 1;
1137
1138 -- 35. Get Feedback on Delivery Agents
1139 SELECT SUM(dr.rating) AS average_rating
1140 FROM Delivery_Review dr
1141 WHERE dr.DeliveryAgent_ID = 1;
1142
1143 -- 36. Get Feedback on Products
1144 SELECT SUM(pr.rating) AS average_review
1145 FROM Product_Review pr
1146 WHERE pr.Product_ID = 1;
1147
1148 -- 37. Get Feedback on Delivery Agents
1149 SELECT SUM(dr.rating) AS average_rating
1150 FROM Delivery_Review dr
1151 WHERE dr.DeliveryAgent_ID = 1;
1152
1153 -- 38. Get Feedback on Products
1154 SELECT SUM(pr.rating) AS average_review
1155 FROM Product_Review pr
1156 WHERE pr.Product_ID = 1;
1157
1158 -- 39. Get Feedback on Delivery Agents
1159 SELECT SUM(dr.rating) AS average_rating
1160 FROM Delivery_Review dr
1161 WHERE dr.DeliveryAgent_ID = 1;
1162
1163 -- 40. Get Feedback on Products
1164 SELECT SUM(pr.rating) AS average_review
1165 FROM Product_Review pr
1166 WHERE pr.Product_ID = 1;
1167
1168 -- 41. Get Feedback on Delivery Agents
1169 SELECT SUM(dr.rating) AS average_rating
1170 FROM Delivery_Review dr
1171 WHERE dr.DeliveryAgent_ID = 1;
1172
1173 -- 42. Get Feedback on Products
1174 SELECT SUM(pr.rating) AS average_review
1175 FROM Product_Review pr
1176 WHERE pr.Product_ID = 1;
1177
1178 -- 43. Get Feedback on Delivery Agents
1179 SELECT SUM(dr.rating) AS average_rating
1180 FROM Delivery_Review dr
1181 WHERE dr.DeliveryAgent_ID = 1;
1182
1183 -- 44. Get Feedback on Products
1184 SELECT SUM(pr.rating) AS average_review
1185 FROM Product_Review pr
1186 WHERE pr.Product_ID = 1;
1187
1188 -- 45. Get Feedback on Delivery Agents
1189 SELECT SUM(dr.rating) AS average_rating
1190 FROM Delivery_Review dr
1191 WHERE dr.DeliveryAgent_ID = 1;
1192
1193 -- 46. Get Feedback on Products
1194 SELECT SUM(pr.rating) AS average_review
1195 FROM Product_Review pr
1196 WHERE pr.Product_ID = 1;
1197
1198 -- 47. Get Feedback on Delivery Agents
1199 SELECT SUM(dr.rating) AS average_rating
1200 FROM Delivery_Review dr
1201 WHERE dr.DeliveryAgent_ID = 1;
1202
1203 -- 48. Get Feedback on Products
1204 SELECT SUM(pr.rating) AS average_review
1205 FROM Product_Review pr
1206 WHERE pr.Product_ID = 1;
1207
1208 -- 49. Get Feedback on Delivery Agents
1209 SELECT SUM(dr.rating) AS average_rating
1210 FROM Delivery_Review dr
1211 WHERE dr.DeliveryAgent_ID = 1;
1212
1213 -- 50. Get Feedback on Products
1214 SELECT SUM(pr.rating) AS average_review
1215 FROM Product_Review pr
1216 WHERE pr.Product_ID = 1;
1217
1218 -- 51. Get Feedback on Delivery Agents
1219 SELECT SUM(dr.rating) AS average_rating
1220 FROM Delivery_Review dr
1221 WHERE dr.DeliveryAgent_ID = 1;
1222
1223 -- 52. Get Feedback on Products
1224 SELECT SUM(pr.rating) AS average_review
1225 FROM Product_Review pr
1226 WHERE pr.Product_ID = 1;
1227
1228 -- 53. Get Feedback on Delivery Agents
1229 SELECT SUM(dr.rating) AS average_rating
1230 FROM Delivery_Review dr
1231 WHERE dr.DeliveryAgent_ID = 1;
1232
1233 -- 54. Get Feedback on Products
1234 SELECT SUM(pr.rating) AS average_review
1235 FROM Product_Review pr
1236 WHERE pr.Product_ID = 1;
1237
1238 -- 55. Get Feedback on Delivery Agents
1239 SELECT SUM(dr.rating) AS average_rating
1240 FROM Delivery_Review dr
1241 WHERE dr.DeliveryAgent_ID = 1;
1242
1243 -- 56. Get Feedback on Products
1244 SELECT SUM(pr.rating) AS average_review
1245 FROM Product_Review pr
1246 WHERE pr.Product_ID = 1;
1247
1248 -- 57. Get Feedback on Delivery Agents
1249 SELECT SUM(dr.rating) AS average_rating
1250 FROM Delivery_Review dr
1
```



Deadline3: SQL



- **Admin Operations:**
 - Top Insights: Identify top 10 customers, products (by revenue and order count), and delivery agents (by rating).
 - **Analytics:** Track average order cost and monthly revenue; monitor stock with low inventory alerts.
 - **Inventory & Product Management:** Update product details and stock levels and maintain product price history.
- **Customer Operations:**
 - **Search & Browse:** Filter products by name, brand, or category.
 - **Cart Management:** Add, remove, update quantity, and calculate total cart value.
 - **Ordering:** Place orders directly from cart; view history and track order status.
 - **Profile & Feedback:** Update personal details, leave product reviews
- **Delivery Agent Operations:**
 - **Order Operations:** Take order, view pending order, delivering history
 - **Performance Tracking:** Monthly delivery count, average ratings, and customer feedback.
 - **Profile:** View and update personal informations

Refinement: ER Diagram

Improved ER Diagram: Smarter & More Scalable Design

This updated ER model is a significant improvement over our earlier version, designed for better normalization, flexibility, and role-based operations.

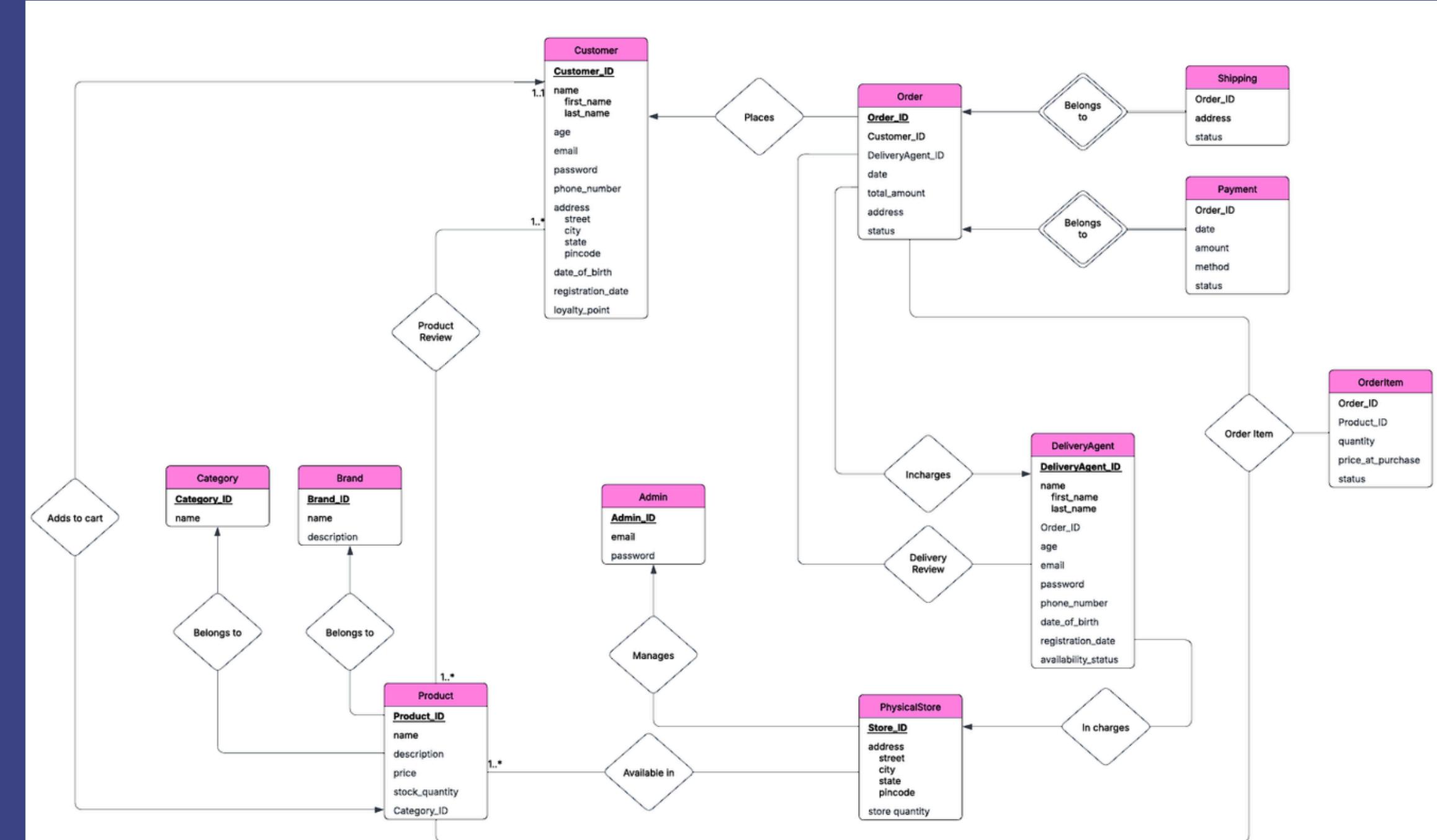
Customers can now easily browse products (by brand or category), manage carts, place orders, and leave reviews.

The Order entity is enhanced with connections to Payment, Shipping, and OrderItem, making it easier to track the complete order lifecycle.

We've introduced DeliveryAgent integration, where agents are auto-assigned and reviewed post-delivery—supporting automation through triggers.

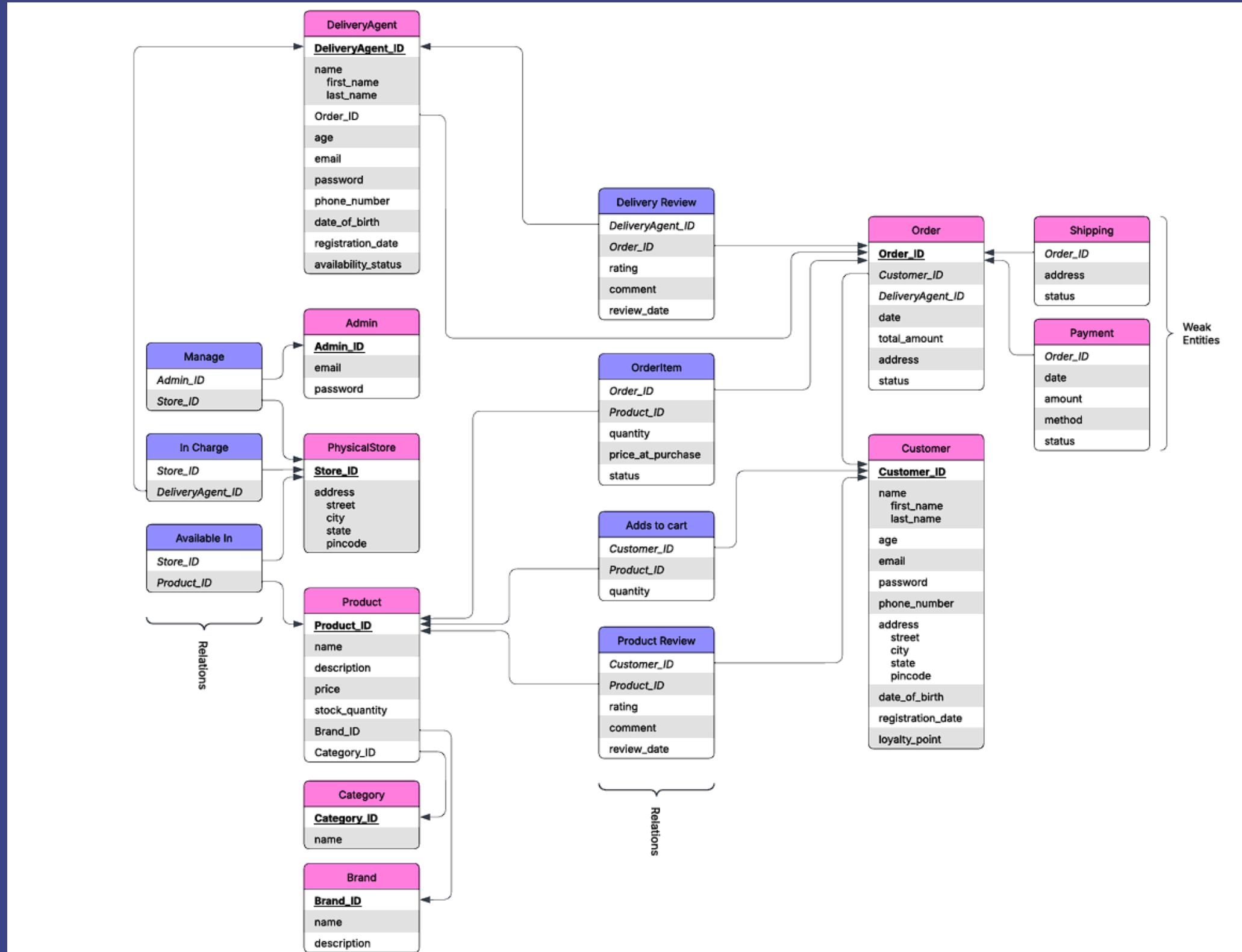
A major enhancement is the store-wise inventory system via the Available_in relation, enabling Admins to manage stock per store instead of globally.

The model clearly separates Brand and Category for products, improving search functionality and analytics.





Refinement: Relational Model



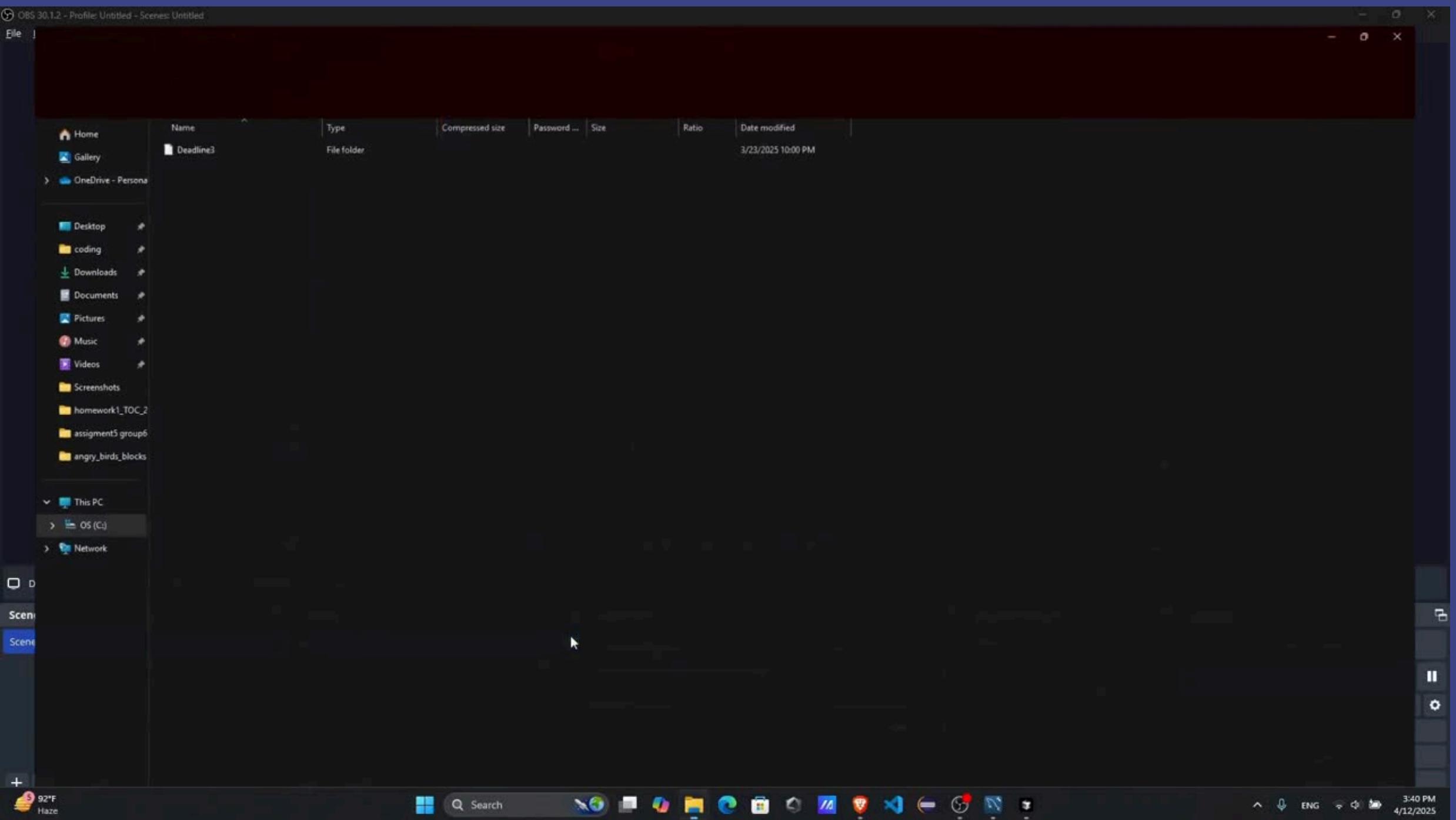
Refined Relational Schema: Clean Structure & Improved Relationships

This updated relational schema builds upon our earlier ER diagram, translating it into a well-structured, normalized relational model. Every entity is now mapped to a relation with clear foreign key connections, ensuring data integrity and referential consistency.

- Customer, Order, and Product remain at the core, now linked via junction tables like OrderItem, AddsToCart, and ProductReview, allowing many-to-many relationships to be handled efficiently.
- Weak entities like Shipping and Payment depend on Order_ID as foreign keys, capturing one-to-one optional relationships in a compact format.
- Roles are separated more clearly: Admins manage stores, and DeliveryAgents are linked to orders and reviewed post-delivery.
- The new design introduces better role-based access and a normalized AvailableIn table to track product availability per physical store, improving inventory handling.



Refinement: Database SQL

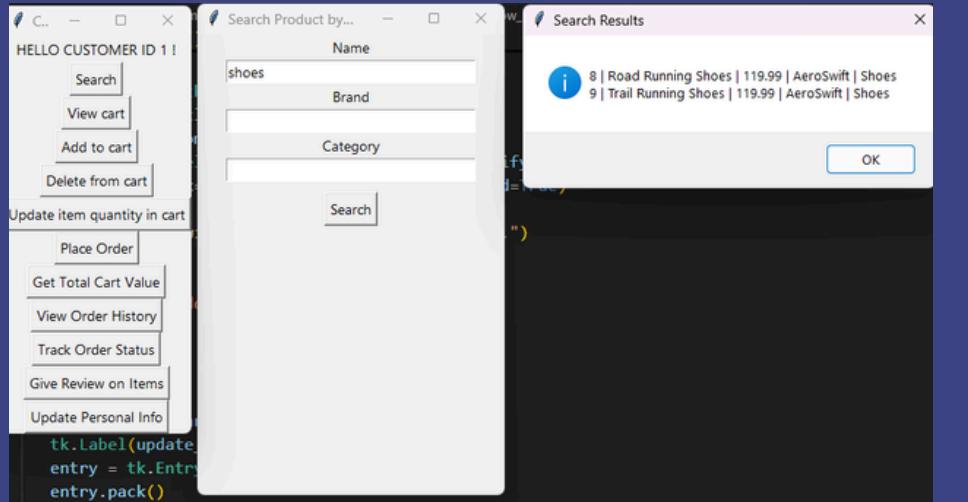


Due to the changes required for Deadline 3, we have updated our ER and relational models, along with the corresponding SQL code. These modifications were necessary to enhance data integrity, normalization, and overall system efficiency. To ensure clarity and transparency in our progress, we have decided to present the revised models and updated implementation in this demo video and through additional slides. This will provide a clearer understanding of the improvements made and demonstrate how the current design better aligns with the project's evolving requirements and real-world functionality.

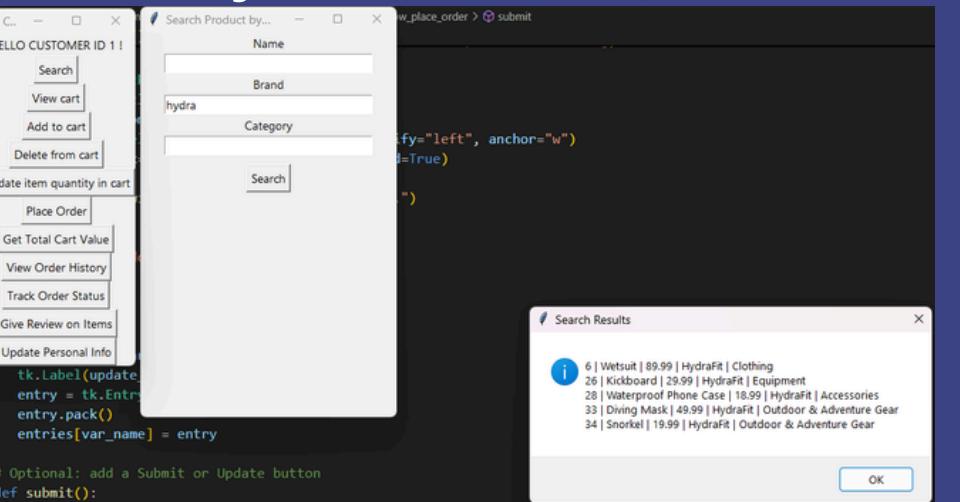
Deadline 4: Customer UI



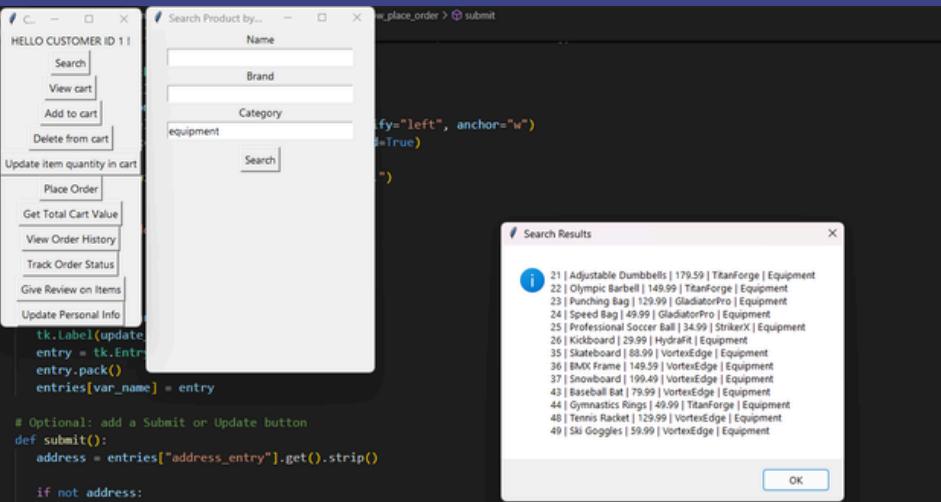
Search by Name



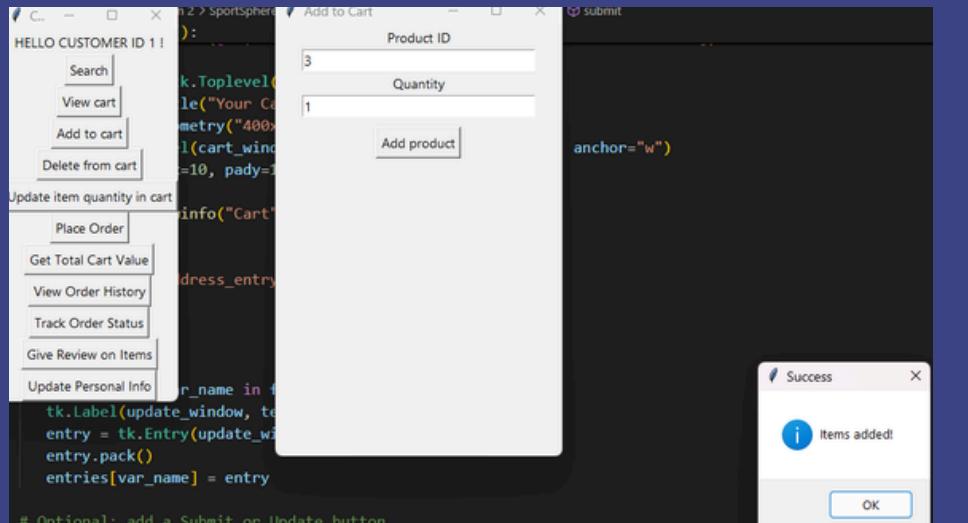
Search by Brand



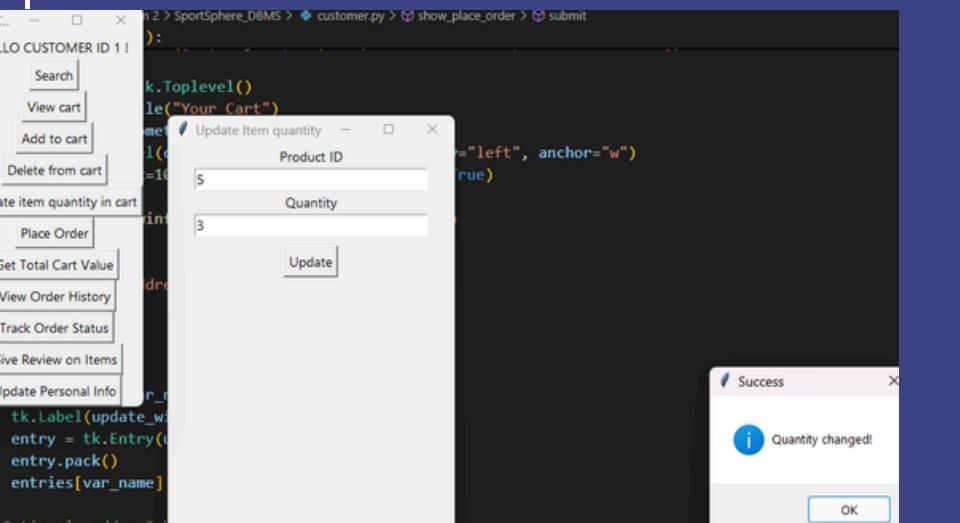
Search by Category



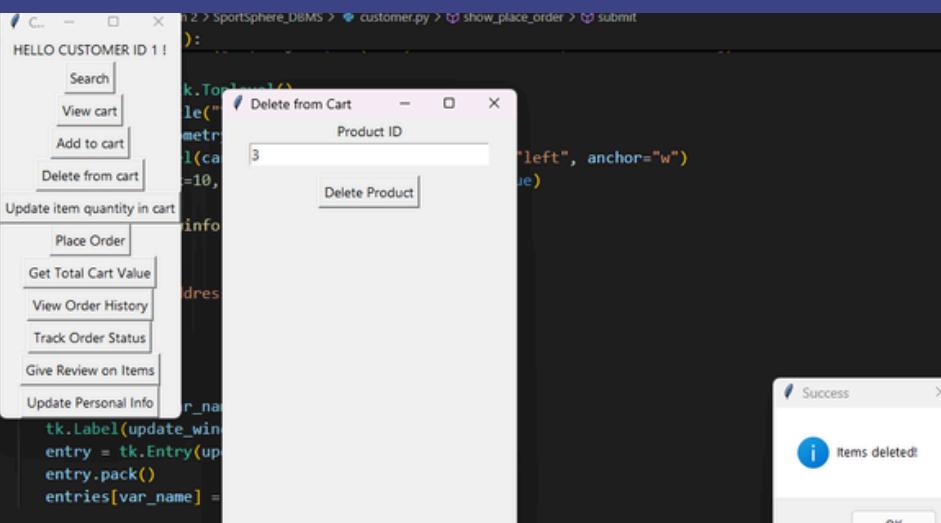
Add Item to Cart



Update Item in Cart



Delete Product from Cart





Deadline 4: Customer UI

View Personal Info

```
customer.py x admin.py sportsphere_functional_queries.sql sportsphere_db.sql
sem 2 > SportSphere_DBMS > Deadline4 > customer.py > view_customer_info
over():

    , fetch=False, params=(address,))

    """
    FROM Adds_to_cart
Customer_ID = %s;
h=False, params=(CUSTOMER_ID,))
x.showinfo("Success", "Order placed!")

    tion as e:
x.showError("Error", f"Something went wrong: {e}")

    e_window, text="Place Order", command=submit).pack(pady=10)

info():
y("SELECT * FROM DeliveryAgent WHERE DeliveryAgent_"
    messagebox.showinfo("My Details", str(rows[0]))


root = tk.Tk()
root.title("Customer Interface")
tk.Label(root, text="HELLO CUSTOMER ID " + str(CUSTOMER_ID) + " !").pack()

    i(1, 'Rahul', 'Sharma', 69, 'rahul.sharma@delivery.com', 'rahulP
    '9876543210', datetime.date(1996, 8, 14), datetime.date(2023
    1), 'Busy')

    C
```

[View Cart](#)

et Cart Value

```
12 > SportSphere_DBMS > customer.py > show_place_order > submit
()):
    k.Toplevel()
    le("Your Cart")
    metry("400x300")
    l(cart_window, text=result, justify="left", anchor="w")
    =10, pady=10, fill="both", expand=True)
    te item quantity in cart
    Place Order
    info("Cart", "Your cart is empty.")

    Get Total Cart Value
    View Order History
    Track Order Status
    Give Review on Items
    update Personal Info
    r_name in fields:
        tk.Label(update_window, text=label_text).pack()
        entry = tk.Entry(update_window, width=30)
        entry.pack()
        entries[var_name] = entry

    Cart value X
    i 89.98
```

lace Order

The screenshot shows a Python application window titled "Your Cart" containing a list of items. The first item is "5 | Goalkeeper Jersey | 3". To the left of the main window, there is a sidebar with various buttons: "Search", "View cart", "Add to cart", "Delete from cart", "item quantity in cart", "Place Order", "Total Cart Value", "New Order History", "Check Order Status", and "Review on Items".

A second window titled "Review Item" is open, showing an address entry field with "n1 boys hostel" and a "Place Order" button.

A third window titled "Success" is displayed, showing a blue circular icon with a checkmark and the text "Order placed!" followed by an "OK" button.

Review Product

The screenshot shows a Python application window titled "customer.py" with the following details:

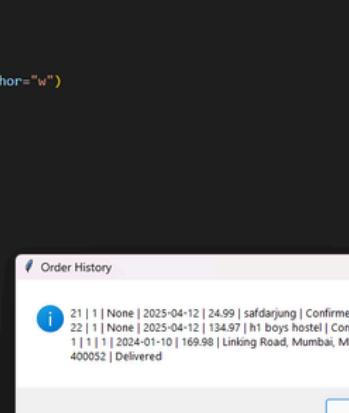
- Customer Interface (Left Window):**
 - Header: HELLO CUSTOMER ID 1!
 - Buttons:
 - Search
 - View cart
 - Add to cart
 - Delete from cart
 - Text: Update item quantity in cart
 - Buttons:
 - Place Order
 - Get Total Cart Value
 - View Order History
 - Track Order Status
 - Give Review on Items
 - Update Personal Info
 - Code snippet:

```
    r_name in fields:  
        tk.Label(update_window, text=label_text).  
        entry = tk.Entry(update_window, width=30)  
        entry.pack()  
        entries[var_name] = entry
```
- Review Item Dialog (Top Right):**
 - Product ID: 3
 - Rating: 1
 - Review: too bad
 - Buttons:
 - Add Review
- Success Dialog (Bottom Right):**
 - Icon: Information
 - Text: Success
 - Text: Review added!

Order History

```
2 SportSphere_DBMS > customer.py > show_place_order > submit
HELLO CUSTOMER ID 1 !
):
    Search
    View cart
    Add to cart
    Delete from cart
Update item quantity in cart
Place Order
Get Total Cart Value
View Order History
Track Order Status
Give Review on Items
Update Personal Info
    for var_name in fields:
        tk.Label(update_window, text=label_text).pack()
        entry = tk.Entry(update_window, width=30)
        entry.pack()
        entries[var_name] = entry

# Optional: add a Submit or Update button
def submit():
    address = entries["address_entry"].get().strip()



| ID | Customer ID | Date       | Total  | Status                   | Notes          |
|----|-------------|------------|--------|--------------------------|----------------|
| 21 | 1           | None       | 24.99  | Confirmed                | safdarjung     |
| 22 | 1           | None       | 134.97 | Confirmed                | ht boys hostel |
| 1  | 1           | 2024-01-10 | 169.98 | Linking Road, Mumbai, MH | 400052         |
|    |             |            |        |                          | Delivered      |


```



Deadline 4: Agent UI

View Pending Order

```

    1. View Pending Orders
    2. Take an Order
    3. View My Orders
    4. Monthly Deliveries
    5. View Average Rating
    6. Update Personal Details
    7. View My Details
    8. Update Availability
    9. Count Deliveries (Month)
    10. View Feedback

    # 8. Update Availability
    def update_availability():
        top = tk.Toplevel()
        top.title("Update Availability")
        tk.Label(top, text="Set status (Available/Busy/On leave):").pack()
        status_entry = tk.Entry(top)
        status_entry.pack()

```

Take Order

```

    1. View Pending Orders
    2. Take an Order
    3. View My Orders
    4. Monthly Deliveries
    5. View Average Rating
    6. Update Personal Details
    7. View My Details
    8. Update Availability
    9. Count Deliveries (Month)
    10. View Feedback

    # 8. Update Availability
    def update_availability():
        top = tk.Toplevel()
        top.title("Update Availability")
        tk.Label(top, text="Set status (Available/Busy/On leave):").pack()
        status_entry = tk.Entry(top)
        status_entry.pack()

```

Delivery History

```

    1. View Pending Orders
    2. Take an Order
    3. View My Orders
    4. Monthly Deliveries
    5. View Average Rating
    6. Update Personal Details
    7. View My Details
    8. Update Availability
    9. Count Deliveries (Month)
    10. View Feedback

    # 8. Update Availability
    def update_availability():
        top = tk.Toplevel()
        top.title("Update Availability")
        tk.Label(top, text="Set status (Available/Busy/On leave):").pack()
        status_entry = tk.Entry(top)
        status_entry.pack()

```

Personal Information Update

```

    1. View Pending Orders
    2. Take an Order
    3. View My Orders
    4. Monthly Deliveries
    5. View Average Rating
    6. Update Personal Details
    7. View My Details
    8. Update Availability
    9. Count Deliveries (Month)
    10. View Feedback

    # 8. Update Availability
    def update_availability():
        top = tk.Toplevel()
        top.title("Update Availability")
        tk.Label(top, text="Set status (Available/Busy/On leave):").pack()
        status_entry = tk.Entry(top)
        status_entry.pack()

```

Average Rating

```

    1. View Pending Orders
    2. Take an Order
    3. View My Orders
    4. Monthly Deliveries
    5. View Average Rating
    6. Update Personal Details
    7. View My Details
    8. Update Availability
    9. Count Deliveries (Month)
    10. View Feedback

    # 8. Update Availability
    def update_availability():
        top = tk.Toplevel()
        top.title("Update Availability")
        tk.Label(top, text="Set status (Available/Busy/On leave):").pack()
        status_entry = tk.Entry(top)
        status_entry.pack()

```

Status Update

```

    1. View Pending Orders
    2. Take an Order
    3. View My Orders
    4. Monthly Deliveries
    5. View Average Rating
    6. Update Personal Details
    7. View My Details
    8. Update Availability
    9. Count Deliveries (Month)
    10. View Feedback

    # 8. Update Availability
    def update_availability():
        top = tk.Toplevel()
        top.title("Update Availability")
        tk.Label(top, text="Set status (Available/Busy/On leave):").pack()
        status_entry = tk.Entry(top)
        status_entry.pack()

```

Monthly Deliveries

```

    1. View Pending Orders
    2. Take an Order
    3. View My Orders
    4. Monthly Deliveries
    5. View Average Rating
    6. Update Personal Details
    7. View My Details
    8. Update Availability
    9. Count Deliveries (Month)
    10. View Feedback

    # 8. Update Availability
    def update_availability():
        top = tk.Toplevel()
        top.title("Update Availability")
        tk.Label(top, text="Set status (Available/Busy/On leave):").pack()
        status_entry = tk.Entry(top)
        status_entry.pack()

```

View Feedback

```

    1. View Pending Orders
    2. Take an Order
    3. View My Orders
    4. Monthly Deliveries
    5. View Average Rating
    6. Update Personal Details
    7. View My Details
    8. Update Availability
    9. Count Deliveries (Month)
    10. View Feedback

    # 8. Update Availability
    def update_availability():
        top = tk.Toplevel()
        top.title("Update Availability")
        tk.Label(top, text="Set status (Available/Busy/On leave):").pack()
        status_entry = tk.Entry(top)
        status_entry.pack()

```

View Personal Details

```

    1. View Pending Orders
    2. Take an Order
    3. View My Orders
    4. Monthly Deliveries
    5. View Average Rating
    6. Update Personal Details
    7. View My Details
    8. Update Availability
    9. Count Deliveries (Month)
    10. View Feedback

    # 8. Update Availability
    def update_availability():
        top = tk.Toplevel()
        top.title("Update Availability")
        tk.Label(top, text="Set status (Available/Busy/On leave):").pack()
        status_entry = tk.Entry(top)
        status_entry.pack()

```



Deadline 4: Admin UI

Top 10 Products by revenue

```

Admin Form
Top 10 customer
Top 10 product by revenue
Top 10 Delivery Agents by Rating
Average Order Cost, Total Number of Orders, Revenue per Month
Top 10 Most Reviewed Products
Low Stock Alert (Under 10)
Update Product
Update Store Stock
SET name = %s
WHERE Product_ID = %s;
""" , fetch=False, params=(name, product_id))

if desc:
    run_query("""
        UPDATE Product
        SET description = %s
        WHERE Product_ID = %s;
    """ , fetch=False, params=(desc, product_id))

if price:

```

Top 10 Delivery Agents

```

Admin Form
Top 10 customer
Top 10 product by revenue
Top 10 Delivery Agents by Rating
Average Order Cost, Total Number of Orders, Revenue per Month
Top 10 Most Reviewed Products
Low Stock Alert (Under 10)
Update Product
Update Store Stock
SET name = %s
WHERE Product_ID = %s;
""" , fetch=False, params=(name, product_id))

if desc:
    run_query("""
        UPDATE Product
        SET description = %s
        WHERE Product_ID = %s;
    """ , fetch=False, params=(desc, product_id))

if price:

```

Top 10 Most Reviewed Products

```

Admin Form
Top 10 customer
Top 10 product by revenue
Top 10 Delivery Agents by Rating
Average Order Cost, Total Number of Orders, Revenue per Month
Top 10 Most Reviewed Products
Low Stock Alert (Under 10)
Update Product
Update Store Stock
SET name = %s
WHERE Product_ID = %s;
""" , fetch=False, params=(name, product_id))

if desc:
    run_query("""
        UPDATE Product
        SET description = %s
        WHERE Product_ID = %s;
    """ , fetch=False, params=(desc, product_id))

if price:

```

Monthly Sales Analytic

```

Admin Form
Top 10 customer
Top 10 product by revenue
Top 10 Delivery Agents by Rating
Average Order Cost, Total Number of Orders, Revenue per Month
Top 10 Most Reviewed Products
Low Stock Alert (Under 10)
Update Product
Update Store Stock
SET name = %s
WHERE Product_ID = %s;
""" , fetch=False, params=(name, product_id))

if desc:
    run_query("""
        UPDATE Product
        SET description = %s
        WHERE Product_ID = %s;
    """ , fetch=False, params=(desc, product_id))

if price:

```

Top 10 Customers

```

Admin Form
Top 10 customer
Top 10 product by revenue
Top 10 Delivery Agents by Rating
Average Order Cost, Total Number of Orders, Revenue per Month
Top 10 Most Reviewed Products
Low Stock Alert (Under 10)
Update Product
Update Store Stock
SET name = %s
WHERE Product_ID = %s;
""" , fetch=False, params=(name, product_id))

if desc:
    run_query("""
        UPDATE Product
        SET description = %s
        WHERE Product_ID = %s;
    """ , fetch=False, params=(desc, product_id))

if price:

```

Low Stock Alert

```

Admin Form
Top 10 customer
Top 10 product by revenue
Top 10 Delivery Agents by Rating
Average Order Cost, Total Number of Orders, Revenue per Month
Top 10 Most Reviewed Products
Low Stock Alert (Under 10)
Update Product
Update Store Stock
SET name = %s
WHERE Product_ID = %s;
""" , fetch=False, params=(name, product_id))

if desc:
    run_query("""
        UPDATE Product
        SET description = %s
        WHERE Product_ID = %s;
    """ , fetch=False, params=(desc, product_id))

if price:

```

Update Store Stock

```

Admin Form
Top 10 customer
Top 10 product by revenue
Top 10 Delivery Agents by Rating
Average Order Cost, Total Number of Orders, Revenue per Month
Top 10 Most Reviewed Products
Low Stock Alert (Under 10)
Update Product
Update Store Stock
SET name = %s
WHERE Product_ID = %s;
""" , fetch=False, params=(name, product_id))

if desc:
    run_query("""
        UPDATE Product
        SET description = %s
        WHERE Product_ID = %s;
    """ , fetch=False, params=(desc, product_id))

if price:

```

Update Product Detail

```

Admin Form
Top 10 customer
Top 10 product by revenue
Top 10 Delivery Agents by Rating
Average Order Cost, Total Number of Orders, Revenue per Month
Top 10 Most Reviewed Products
Low Stock Alert (Under 10)
Update Product
Update Store Stock
SET name = %s
WHERE Product_ID = %s;
""" , fetch=False, params=(name, product_id))

if desc:
    run_query("""
        UPDATE Product
        SET description = %s
        WHERE Product_ID = %s;
    """ , fetch=False, params=(desc, product_id))

if price:

```



Thank You For Watching

Tung Duc Vu (2023558), Gaurav Shankar (2023219), Aditya Chaudhary (2023042), Turbold Amarbat (2023559)